Dynamic path planning for collision avoidance in a robotized framework for autonomous driving verification

Daniel Johansson

LUND UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6087
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Printed in Sweden by Tryckeriet i E-huset
Lund 2019

Abstract

Self-driving vehicles is a highly anticipated technology for increasing the safety and efficiency of automotive transportation systems by removing the risk of human errors. Volvo Car Corporation is determined to produce vehicles of full autonomy and highest safety within the near future. To achieve this goal, Volvo Cars is in parallel with the development of autonomous vehicles setting up a sophisticated pipeline for verifying and testing the autonomous driving functions. This thesis revolves around the last step of this pipeline, by implementing and further developing an algorithm for collision avoidance in a robotized framework for verification of autonomous driving where the functions are tested on real vehicles on a test track.

The proposed algorithm used to achieve collision avoidance in the robotized test framework is the *Bicycle Optimal Reciprocal Collision Avoidance* (B-ORCA) algorithm. This algorithm uses a construct called *Velocity Obstacle* to predict imminent collisions between vehicles in a scenario and then calculates the optimal velocities to avoid the collisions in a collaborative manner.

To evaluate the performance of the algorithm, a set of experiments were performed on the driving robots that will be used in the testing framework, both in simulations and on a real vehicle. The results from these experiments show that the current implementation of the B-ORCA algorithm guarantees accurate safe trajectories up to speeds of 50km/h. To support speeds above 50km/h, the simple *Kinematic bicycle model* currently used to calculate the trajectories has to be replaced with a more sophisticated motion model. This new model has to better model the lateral acceleration that, with too high values, was shown to be the main parameter that made the vehicle not follow the safe trajectories as desired.

Acknowledgements

I would first like to thank Siddhant Gupta and Francesco Costagliola at Volvo Car Corporation for providing me with this Master's Thesis opportunity, and for their guidance and feedback throughout this journey. I would also like to express my deepest gratitude to my main supervisor Angel Molina Acosta at Volvo Car Corporation for his continuous support and guidance throughout the whole thesis study. His dedication and interest to this project have been very inspiring. I would also like to show my greatest appreciation for Salma Abdelwahab, doing her Master's Thesis in the same project, for valuable cooperation, discussions and input throughout the thesis. I would like to thank my academic supervisor Prof. Anders Robertsson for valuable inputs and suggestions. Lastly, I would like to extend my gratitude to the Virtual Car teams on Volvo Car Corporation for giving support on technicalities and simulation tools, and the Robot team at Hällered Proving Grounds for helping me perform experiments on the test track.

Daniel
Gothenburg, June 2019

Contents

1

Introduction

1.1 Motivation

The field of autonomous vehicles has gained great attention the latest years by both the automotive industry and academia. Great advancements within computer and sensor development allows powerful computers and sensors to be installed directly in the vehicles, which opens up doors for advanced perception and control algorithms to be run in real time. Driving vehicles is statistically a dangerous way of transportation. Every year, 20-50 million people are injured and 1.25 million die globally from road accidents, making road accidents the 9th leading cause of death and account for 2.2% of all deaths [Association for safe international road travel, 2018]. Safe road transportation is therefore very important and automotive companies are focusing a lot of research on making their vehicles safer.

Throughout the history of automotive transportation, there have been some revolutionary innovations which made it much safer to travel by road. In 1959, the three-point seatbelt was developed by Volvo Cars which greatly reduced the risk of severe injuries in traffic collisions. With its simple design, the three-point seatbelt is very easy to put on, and still effectively protects both the upper and lower body of the passenger. Due to its significance, Volvo Cars soon released the patent for other car manufacturers to use and it is today standard in all vehicles [Volvo Car Corporation, 2009]. The airbag is another invention that has helped saving many lives in road accidents. Rudimentary patents on the airbags goes back to the 1950s [Bellis, 2018], but it was not until the 1980s that it was available to customers when Ford and Chrysler introduced them in their vehicles and included them as standard equipment in 1990 [McCormick, 2019].

Using control systems to reduce road accidents was first introduced with the Anti-Lock Braking System (ABS) in 1978. The ABS prevents the wheels to lock in hard brakes, which allows the driver to still maintain control of the vehicle and at the same time reduce the braking distance, reducing the risk of severe crashes [Burton et al., 2004]. Other control systems for active safety has since been developed and included in modern vehicles. One of the most notable of these systems is ESC (Electronic Stability Control) that prevents instability that leads to spinning

by detecting loss of traction and applies brakes on individual wheels to "steer" the vehicle in the right direction [Euro NCAP, 2019].

Still, with all the safety functions installed in modern vehicles, there are a lot of driver induced crashes leading to severe or fatal outcomes. One of the leading reasons for deaths in traffic is driving intoxication, which in Sweden was responsible for 32 percent of all fatal accidents in 2017 [Trafikverket, 2018b]. In fact, most accidents in the traffic are caused by the driver and human error. In 1997 the Swedish government together with the Swedish Transport Administration introduced the *Vision Zero* [Trafikverket, 2018a], with the goal that no one should die or be severely injured in a traffic accident. There have been many achievements along the way and the number fatal accidents has been reduced, but they have not disappeared completely as the human error still has an impact on the safety of the transportation system. Autonomous driving could therefore be the technology that achieves the goals of the *Vision Zero* as it removes the human error from the traffic system.

Autonomous driving is today a common topic in mainstream media and often focused on the achievements done by the large tech-companies. For example, Tesla announced recently that all their newly produced vehicles will have the hardware needed for autonomous driving and released promising footage of their currently available auto-pilot system [Tesla, 2016]. Google, Uber and other more conventional car manufacturers are also working on their own projects for autonomous driving [CBS Insight, 2018]. Not all reports about the development of autonomous vehicles are positive. In 2018, a Tesla car driving on autopilot on a Californian highway crashed and killed the passive driver [The Guardian, 2018], and an autonomous testing vehicle from Uber struck and killed a pedestrian in Arizona [New York Times, 2019]. The human error was still the main cause in both accidents, but it remains clear that the safety aspects are regarding the autonomous vehicles of uttermost importance for the society and the companies involved. It is indisputable that a lot of research and verification still remains to be done before fully autonomous vehicles drive the streets.

The research on autonomous driving has come a long way and the first prototypes of the autonomous vehicles are currently being tested, but there is still a long way of research and verification ahead to solve problems related to the safety and the reliability of the autonomous vehicles. This thesis is focusing on the verification of the autonomous driving function. Volvo Car Corporation is building an automatic robotized verification system on the test track for the autonomous vehicles. In this system, several vehicles are controlled with robots to replay a scenario that the autonomous vehicle should handle. With this background, the focus for this thesis is to investigate, further develop and evaluate an algorithm for collision avoidance with multiple agents proposed in [Van Den Berg et al., 2011]. This algorithm will be used if a test on the test track goes wrong and has to end safely without the vehicles crashing into each other. The algorithm uses a construct called *Velocity obstacles* to predict if a collision will occur in the near future and then calculates safe velocities that the vehicles have to adopt to avoid the collision. The algorithm makes use of the

physical constraints of the vehicles and calculates safe trajectories for every vehicle in the test, where every vehicle takes responsibility to avoid the collision.

1.2 Background

In 2014, the Society of Automotive Engineers (SAE) introduced a 6 level standard (0 to 5) for classifying vehicles with autonomous driving capabilities [SAE International, 2018]. The first and most basic is level 0, *No Driving Automation*, where the driver controls all aspects of the car even when enhanced with active safety functions. Level 1, *Driver Assistance*, make the system and user share control of certain aspects of the vehicles. For example, Adaptive cruise control and ESC systems are classified as level 1. More advanced lateral and longitudinal control systems, as for example emergency braking, are part of level 2 *Partial Driving Automation*. When a vehicle is classified as level 3 of autonomy, *Conditional Driving Automation*, it is capable of full autonomous driving in certain conditions and full fallback on the driver if anything outside of the system's capabilities occurs. Vehicles of level 4, *High Driving Automation*, is capable of full autonomous driving in certain conditions with with no fallback help from the driver. Level 5 vehicles of autonomous drive, *Full Driving Automation*, is capable of handling all conditions and environments with no help of the driver at all.

Volvo Car Corporation has during previous years developed several functions classified at level 1 and 2 to increase the safety related to their cars. These functions are still very important, but Volvo Cars is now also focusing much resources on producing cars that apply autonomous driving capabilities classified at level 4 and 5. i.e. the highest levels for autonomous vehicles. This is no trivial task, and in parallel with the development of the full autonomous cars a sophisticated system for testing and verification has to be in place to guarantee the safety of the autonomous cars. Volvo Cars is therefore building a whole verification system for testing the autonomous vehicles before they are released to the public. This verification is done in four steps; Model-in-the-loop (MIL), software-in-the-loop (SIL), hardware-in-the-loop (HIL) and vehicle-in-the-loop (VIL). This thesis revolves around developing functions for the system for VIL, which is done on the test track with all sensors and software installed in the vehicle under test (VUT). The environment surrounding the VUT is replaying a scenario with real physical vehicles as well as injected virtual vehicles of high risk targets. The real vehicles on the test track will be driven by specialized driving robots, which take full control of the steering and acceleration of the car to navigate it according what is decided on a centralized server. This system also allows *soft targets* that represent for example pedestrians and animals to be included in the scenario. All objects in the system are controlled by a centralized server and the control signals, measurements and relevant information are sent over a 4G network. See Figure 1.1 for a simplified overview of the system.

The development of this system for vehicle in the loop verification is done in col-

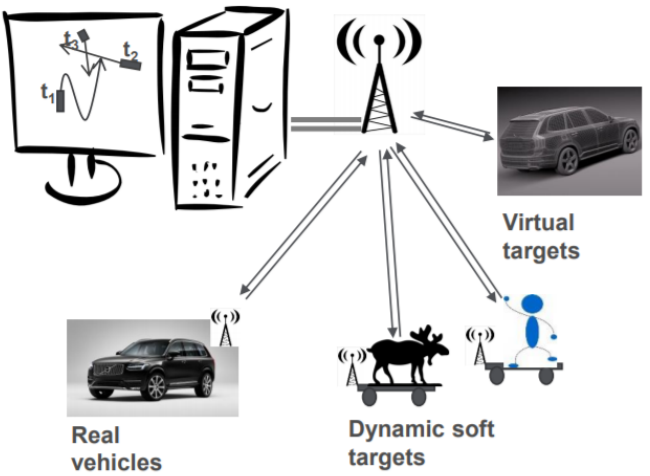

Figure 1.1 Simplified overview of the Vehicle in the loop verification system. This system includes real vehicles controlled by driving robots, and soft targets and is connected to a centralized server. This server runs the verification tests and has also the possibility to inject virtual vehicles into the system.

laboration with other partners in the CHRONOS2 project. The goal of this project is to produce a functioning prototype of a VIL verification system. The partners in this project also exchange valuable ideas related to the project which is helpful for the work done internally by the partners themselves.

1.3 Scope

Aim and Delimitations

The focus for this thesis is to produce an algorithm that calculates safe and collision-free trajectories for multiple vehicles in a cooperative manner given their current initial states. For this task, *Bicycle Optimal Reciprocal Collision Avoidance* (B-ORCA) [Alonso-Mora et al., 2012] is the proposed algorithm. The purpose for this algorithm is to always guarantee safe exit ways for testing vehicles used for autonomous driving verification where all vehicles are controlled from one singular entity. The safe exit trajectories will therefore be used as backup trajectories the vehicles can fall back to if anything in the test goes wrong. The algorithm is therefore not meant to be used to achieve full autonomous driving, as autonomous vehicles operate as independent agents in a large traffic system.

The question of when the safe trajectories should be activated is very relevant in order to guarantee collision free tests, as due to the motion constraints of the vehi-

cles, it may exist scenarios where a collision is physically impossible to avoid. This problem formulation is not covered in this thesis, but is researched in another thesis work [Abdelwahab, 2019] run in parallel to this thesis at Volvo Car Corporation, as it is very relevant to trigger the safe trajectories before the viable safe trajectories cease to exist.

Research Questions

This thesis revolves around answering the following research questions:

- How well does the proposed optimal dynamic trajectory planning algorithm for collision avoidance perform when used on driving robots?
- Is a simple model describing the vehicle's motion enough to generate accurate and drivable trajectories?

By answering the above stated questions, this thesis will contribute to the research done at Volvo Car Corporation and in the CHRONOS2 project.

1.4 Outline

Chapter 2 covers the full theoretical background the proposed algorithm is based upon, describing how eventual future collisions can be detected and avoided. Chapter 3 covers the implementation of the proposed algorithm, ranging from the framework the algorithm used for implementation, and a full description of the algorithm. In Chapter 4, the experimental procedure of how the verification of the performance will be done is thoroughly described. The verification is done by defining a set of benchmarked dangerous scenarios where the proposed algorithm will make the involved vehicles avoid the imminent collisions. In Chapter 5, the results from the simulations and tests run in the experiments are presented, together with an average statistical performance measure of the algorithm. These results are discussed in Chapter 6 and the most important findings are concluded in Chapter 7 where also future work and possible improvements of the algorithm are brought up.

1.5 Related work

The main subject of this thesis revolves around trajectory planning and collision avoidance for autonomous vehicles. An in-depth study of the current state-of-the-art algorithms for trajectory planning is covered in [González et al., 2016]. One of the most prominent of these algorithms is the *A* algorithm* [Hart et al., 1968], which is a heuristic graph search algorithm closely related to *Dijkstra's algorithm* [Dijkstra, 1959]. The *A** algorithm can also be extended with a non-holonimc motion model to be applied on vehicle-like agents in continuous space [Petereit et al., 2012]. The

*A** algorithm is though limited in its use in environments with moving obstacles as it assumes static maps. Therefore, the alternative algorithm *D** (*Dynamic A**) was introduced to use similar principles in dynamic maps [Stentz, 1994]. This algorithm has also been further developed for calculation efficiency and generality [Likhachev et al., 2001].

Another popular family of trajectory planners are the *Sampling based planners*, where the algorithms performs random sampling of a state space to get to a desired location. The most notable algorithms of this family are the *Probabilistic Roadmap Method (PRM)* [Kavralu et al., 1996] and the *Rapidly-exploring Random Tree (RRT)* [LaValle, 1998]. PRM is commonly used in robotics, and RRT has on later years been extensively tested for autonomous vehicles, by amongst other the MIT team at DARPA Urban Challenge [Kuwata et al., 2009]. The main drawbacks with these planners are that they only produce suboptimal solutions.

The common goal for trajectory planning algorithms is to find the best way from a starting position to a desired goal in a known, or semi-known environment. These algorithms do therefore not try to predict collisions of different moving objects beforehand, which makes them not optimal for scenarios where collision avoidance is of first priority. Alternative algorithms that will implement collision avoidance more efficiently is to use a *MPC* planner [Ghazaei Ardakani et al., 2015] or a planner based on *Artificial Potential fields (APF)* [Khatib, 1985]. The MPC is very good in calculating optimal trajectories, but the calculations are very heavy and might take much time, and every configuration of a MPC is specialized for a specific scenario, making it non-flexible for more generalized use. The APF planner is not as computational heavy as the MPC, but has problems with local minimum that can cause problem calculating collision free trajectories. There are extensions to the APF planner that address the local minimum problem, but there is yet no final solution to it [Zhou and Li, 2014].

The *Optimal Reciprocal Collision Avoidance (ORCA)* algorithm [Van Den Berg et al., 2011] used for collision avoidance in this thesis revolves around the concept of velocity obstacles [Van den Berg et al., n.d.] to predict and avoid collisions reciprocally. The algorithm has proven to be effective in avoiding collision of both a small number of agents and of larger swarms of agents, and it can be applied on any configuration in the state-space. When the ORCA algorithm is extended with a non-holonomic vehicle model it is also possible to apply its principles to vehicle-like agents to avoid collisions in effective manners [Alonso-Mora et al., 2012]. To use the ORCA algorithm to make non-holonomic vehicles come to a quick and safe stop, which this thesis focuses on, has not been found to have been investigated in any other work before.

2

Theory

This chapter presents the background theory of the proposed B-ORCA algorithm. First, the original *Optimal Reciprocal Collision Avoidance* (ORCA) algorithm is thoroughly described to show its fundamental functions in Section 2.1. After that, the ORCA algorithm is extended with a non-holonomic motion model and forms the *Bicycle Optimal Reciprocal Collision Avoidance* (B-ORCA) algorithm, with its components described in Section 2.2.

2.1 Optimal Reciprocal Collision Avoidance

Optimal Reciprocal Collision Avoidance (ORCA) is an algorithm for collision avoidance with n number of agents which was first proposed in [Van Den Berg et al., 2011]. It is a centralized algorithm that calculates collision-free trajectories for every agent that can be controlled. To do this, the algorithm makes use of so called *Velocity Obstacles* (VOs) [Fiorini and Shiller, 1998] to calculate collision-free trajectories within a given time horizon. For the agent to not collide with the other object it therefore has to choose a velocity that stays outside of this VO. How the VO is constructed is covered the next subsection.

The collision avoidance becomes even more efficient if all controllable agents cooperate to avoid the collision. This is done by letting both actors *reciprocally* contribute to escaping the VO.

When a potential future collision is detected based on a VO, a safe set of velocities is given that an agent can take to avoid the collision. As the agent then has an *optimization velocity* $\mathbf{v}^{opt}$ that it wants to follow, an optimization problem is solved to find the velocity closest to $\mathbf{v}^{opt}$. By doing this optimization, it is possible to take into account constraints from all surrounding agents and obstacles to avoid all of them and also follow a desired path enforced by $\mathbf{v}^{opt}$.

In the following sub-chapters a more detailed description of the ORCA algorithm will be given to present the functionality of the algorithm. The original version of the ORCA assumes that the agent is *holonomic*, which means that the agent can move freely in any direction. In the case of this thesis, the agents are vehicles

which are *non-holonomic* and do not have the possibility to move freely as they have to follow the corresponding vehicle model. An extension of the ORCA algorithm, the *Bicycle Optimal Reciprocal Collision Avoidance* (B-ORCA) presented in [Alonso-Mora et al., 2012], is therefore used to generate trajectories that are viable for vehicles to follow.

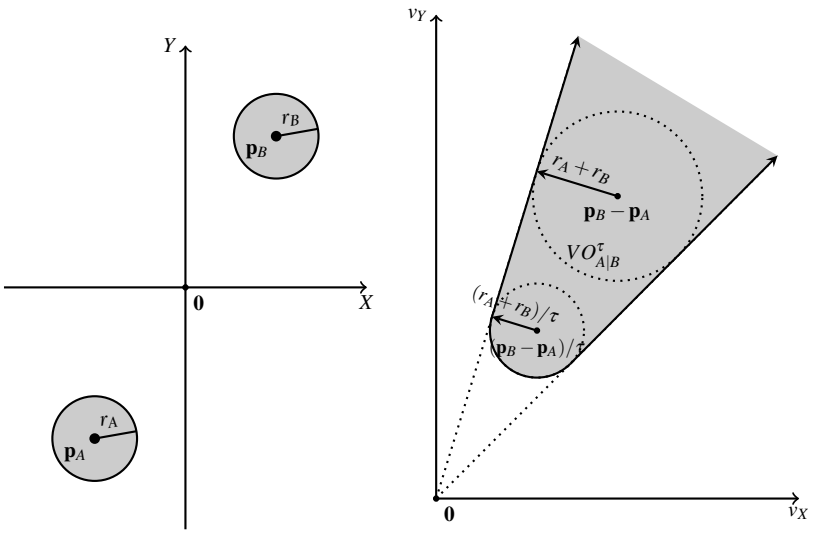

Figure 2.1 **(Left)** The agents position in a x,y-plane specified with p_i and their safe boundaries specified by r_i. **(Right)** The constructed Velocity Obstacle for object A in relation to object B. If a velocity is selected for object A that is outside of $VO^{\tau}_{A|B}$ (dark grey area), a collision-free trajectory is guaranteed for at least time τ. Figure adapted from [Van Den Berg et al., 2011].

Velocity Obstacle

As defined by [Van Den Berg et al., 2011], a Velocity Obstacle for an agent A induced by B is the set of relative velocities that will make the objects collide within a specified time τ. It is formally written as $VO^{\tau}_{A|B}$. A visual representation of the Velocity Obstacle is shown in Figure 2.1. To construct the $VO^{\tau}_{A|B}$ the following information needs to be know about the objects:

- p_A and p_B: The positions of object A and B.
- r_A and r_B: The radii of the safe boundaries of the object A and B.
- $\mathbf{v}_A$ and $\mathbf{v}_B$: The current velocities for object A and B.

- τ: The look-ahead time for which a collision should be avoided.

With this data, the Velocity Obstacle $VO^{\tau}_{A|B}$ is constructed according to Figure 2.1. If the relative velocity of agent A and B lies inside $VO^{\tau}_{A|B}$ a collision will occur within the time τ. This can formally be written as $\mathbf{v}_A - \mathbf{v}_B \in VO^{\tau}_{A|B}$, and conversely $\mathbf{v}_B - \mathbf{v}_A \in VO^{\tau}_{B|A}$.

The definition of the Velocity Obstacle induced by the other agent B, $VO^{\tau}_{A|B}$, can formally be written as:

$$VO^{\tau}_{A|B} = \{\mathbf{v} | \exists t \in [0,\tau], t\mathbf{v} \in D(\mathbf{p}_B - \mathbf{p}_A, r_A + r_B)\}. \tag{2.1}$$

This expression should be read as: The union of the set of velocities inside the disks D with radius $r_A + r_B$ and center in $\mathbf{p}_B - \mathbf{p}_A$, with an added scaling factor $t \in [0,\tau]$.

Collision Avoidance

If a future potential collision has been detected with a Velocity Obstacle, the ORCA algorithm will calculate the appropriate actions to avoid the collision. This is done by calculating the *shortest* way out of the Velocity Obstacle, e.g. the vector from $\mathbf{v}^{opt}_B - \mathbf{v}^{opt}_A$ to the closest point of the boundary of the Velocity Obstacle.[1] This vector will be denoted $\mathbf{u}$ and is visualized in Figure 2.2.

$$\mathbf{u} = (\underset{\mathbf{v} \in \delta VO^{\tau}_{A|B}}{\arg\min} \|\mathbf{v} - (\mathbf{v}^{opt}_A - \mathbf{v}^{opt}_B)\|) - (\mathbf{v}^{opt}_A - \mathbf{v}^{opt}_B). \tag{2.2}$$

The vector $\mathbf{u}$ is therefore describing the smallest change of the relative velocity that is needed to avoid the collision between agent A and B. To share the responsibility of avoiding the collision, each agent will adapt $\frac{1}{2}\mathbf{u}$ and then assume that the other agent will take care of the other half. Lets now denote $ORCA^{\tau}_{A|B}$, which is the set of velocities for agent A that will make it avoid the collision with agent B. This set is geometrically constructed by a straight line with its normal vector pointing in the direction of $\mathbf{n}$ (the outward normal of the boundary of $VO^{\tau}_{A|B}$ at the point $(\mathbf{v}^{opt}_B - \mathbf{v}^{opt}_A) + \mathbf{u}$), starting at the point $\mathbf{v}^{opt}_A + \frac{1}{2}\mathbf{u}$, see Figure 2.2. This is more formally written as:

$$ORCA^{\tau}_{A|B} = \{\mathbf{v} | (\mathbf{v} - (\mathbf{v}^{opt}_A + \frac{1}{2}\mathbf{u})) \cdot \mathbf{n} \geq 0\}. \tag{2.3}$$

If agent A then adopts a velocity within $ORCA^{\tau}_{A|B}$ and agent B adopts a velocity within $ORCA^{\tau}_{B|A}$ the potential collision will be avoided. This is done without the agents communicating, as long as they *observe* each others position, radius, and optimization velocity.

[1] Introducing the *optimization velocities* $\mathbf{v}^{opt}$ to make the definition of ORCA more general. These are most often selected as the agents' current velocities, but can also be selected in alternative ways to change the behavior of the ORCA algorithm.

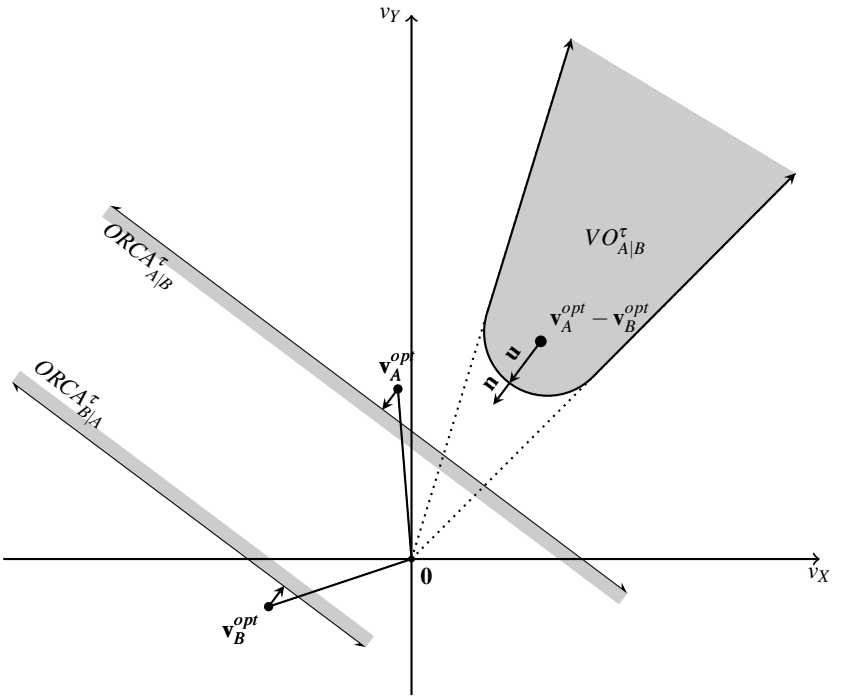

Figure 2.2 Visualization of how the set $ORCA^{\tau}_{A|B}$ is determined. The vector $\mathbf{u}$ is first calculated as the minimum change in the relative velocity that is needed to escape $VO^{\tau}_{A|B}$. $ORCA^{\tau}_{A|B}$ is then determined as the half plane perpendicular to $\mathbf{u}$ that goes through the point $\mathbf{v}_A^{opt} + \frac{1}{2}\mathbf{u}$. Figure adapted from [Van Den Berg et al., 2011, p. 7].

Collision Avoidance with multiple agents

The above mentioned collision avoidance method can be extended to make the agents avoid n-number of other agents. The agent observes the other objects' positions, radii, and optimization velocities and then constructs the Velocity Obstacles to calculate the set of safe velocities $ORCA^{\tau}_{A}$ that will avoid the collisions with the other objects. This safe set can formally be denoted as

$$ORCA^{\tau}_{A} = \bigcap_{A \neq i} ORCA^{\tau}_{A|i} \tag{2.4}$$

where $ORCA^{\tau}_{A|i}$ is the set of velocities for A that will avoid the collision with every other object i, see Figure 2.3.

The agent has a velocity it wants to follow, $\mathbf{v}_A^{opt}$, so the next step will be to select the new velocity $\mathbf{v}_A^{new}$ from $ORCA^{\tau}_{A}$ so it is as close to $\mathbf{v}_A^{opt}$ as possible. This task fits

perfectly in a convex optimization problem, as the set $ORCA_A^\tau$ is bounded by straight lines, which makes it a convex set, and the distance $\|\mathbf{v}_A^{new} - \mathbf{v}_A^{opt}\|$ to be minimized is a convex function.

The definition of a convex problem in standard form is

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq b_i,\ i = 1,\ldots,m. \\ & a^T x = c_i,\ i = 1,\ldots,n. \end{aligned} \tag{2.5}$$

where $f_i(x)$ are convex functions and a, b, and c are vectors [Bertsekas, 2009, p. 127]. In the case of solving the convex optimization problem of finding $\mathbf{v}_A^{new}$, the only convex function is $f_0(x)$ which describes the distance $\|\mathbf{v}_A^{new} - \mathbf{v}_A^{opt}\|$. The functions $f_i(x) \leq b_i,\ i = 1,\ldots,m$ are induced by the half-planes $ORCA_{A|i}^\tau$ which can be written on affine form $a^T x \leq b$.

The convex problem can therefore be reduced to

$$\begin{aligned} \text{minimize} \quad & \|\mathbf{v}_A^{new} - \mathbf{v}_A^{opt}\| \\ \text{subject to} \quad & a_i^T x \leq b_i,\ i = 1,\ldots,m. \end{aligned} \tag{2.6}$$

where the parameters of a_i and b_i are set depending on the where the half-planes $ORCA_{A|i}^\tau$ are located. Solving this problem will give the new velocity $\mathbf{v}_A^{new} \in ORCA_A^\tau$ which is closest to $\mathbf{v}_A^{opt}$. This is visualized in Figure 2.3.

When the new velocity of agent A has been determined the agent will take a step to its new location according to:

$$\mathbf{p}_A^{new} = \mathbf{p}_A + \mathbf{v}_A^{new}\Delta t \tag{2.7}$$

where Δt is defined as the time between every iteration of calculating a safe velocity.

Densely Packed Conditions

In the case of scenarios that are very densely packed with agents and obstacles, a velocity might not exist that satisfies all constraints induced by the half planes $ORCA_{A|i}^\tau$. The set of safe velocities is therefore $ORCA_A^\tau = \emptyset$. To handle these occasions, the ORCA algorithm proposed in [Van Den Berg et al., 2011] instead looks for the *safest* velocity that minimally violates the constraints imposed. Let $d_{A|i}(\mathbf{v})$ be the Euclidean distance from a velocity $\mathbf{v}$ to the edge of the half-plane $ORCA_{A|i}^\tau$. To select this new velocity $\mathbf{v}_A^{new}$, the maximum distance to any of the half-planes should be minimized:

$$\mathbf{v}_A^{new} = \arg\min(\max_{A \neq i} d_{A|i}(\mathbf{v})). \tag{2.8}$$

This minimization can geometrically be interpreted as moving the half-planes $ORCA_{A|i}^\tau$ perpendicular outwards with the same speed, until one velocity becomes

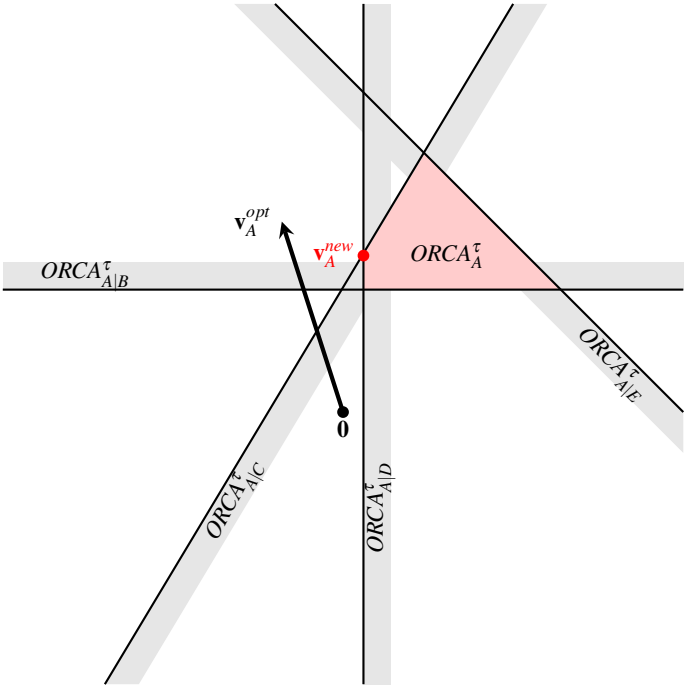

Figure 2.3 The half-planes of safe velocities for agent A induced by each agent in a simulation. To avoid collision with the other agents, agent A has to choose a velocity in the intersection of these half planes. This intersection is the convex set $ORCA_A^\tau$. The new velocity $\mathbf{v}_A^{new}$ inside $ORCA_A^\tau$ is desired to be as close to $\mathbf{v}_A^{opt}$ as possible. $\mathbf{v}_A^{new}$ is therefore calculated by solving a convex optimization problem.

valid. If the convex optimization of the set $ORCA_A^\tau$ is seen as hard constraints that guarantee collision avoidance, the method for finding a velocity $ORCA_A^\tau = \emptyset$ can be seen as imposing soft constraints. This means that all hard constraints should be violated as little as possible.

Static Obstacles

The ORCA algorithm can also make the agents avoid collisions with static obstacles, and not just other agents which has been covered up until this point. The static obstacles are avoided by the moving agents in a very similar way as they avoid each other, as explained next.

As for agent-agent collision avoidance, agent-obstacle collision avoidance also relies on Velocity Obstacles to calculate sets of *safe* velocities the agent has to adopt to avoid collision with the static obstacle. In this framework, a static obstacle is generally modelled as a collection of straight line segments, and as the agent is

modeled as a disk with radius r_A with position $\mathbf{p}_A$ it is possible to model the Velocity Object for the static object segment O as:

$$VO^{\tau}_{A|O} = \{\mathbf{v} | \exists t \in [0,\tau], t\mathbf{v} \in O \oplus -D(\mathbf{p}_A, r_A)\} \tag{2.9}$$

This expression should be read as: the union of the set of velocities inside the Minkowski sums $O \oplus -D(\mathbf{p}_A, r_A)$, with an added scaling factor $t \in [0,\tau]$, see Figure 2.4 for a visualization of $VO^{\tau}_{A|O}$.

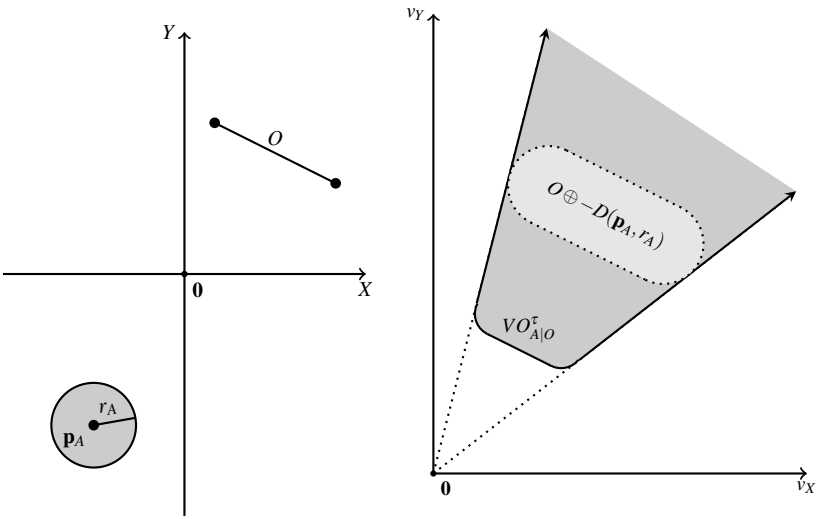

Figure 2.4 **(Left)** The configuration of the agent A and the static obstacle O. **(Right)** The constructed Velocity Obstacle $VO^{\tau}_{A|O}$ from the configuration in (left) with $\tau = 2$. Figure adapted from [Van Den Berg et al., 2011, p. 12].

In the same way as of the agent-agent Velocity Obstacles, agent A will collide onto the static object segment if $\mathbf{v}^{opt}_A \in VO^{\tau}_{A|O}$. To avoid the collision, the change in velocity $\mathbf{u}$ that is needed to escape $VO^{\tau}_{A|O}$ is calculated which will give the *safe* velocity set $ORCA^{\tau}_{A|O}$ as:

$$ORCA^{\tau}_{A|O} = \{\mathbf{v} | (\mathbf{v} - (\mathbf{v}^{opt}_A + \mathbf{u})) \cdot \mathbf{n} \geq 0\} \tag{2.10}$$

Note that the agent A takes full responsibility in avoiding collision as the coefficient in front of $\mathbf{u}$ is 1. This is quite intuitive due to the fact that only the agent should take action in avoiding the collision as the static object can not move. When the $ORCA^{\tau}_{A|O}$ has been calculated, the half-plane is passed to the convex optimization algorithm together with the other half-planes in order to calculate $ORCA^{\tau}_{A}$ in (2.4).

2.2 Bicycle Optimal Reciprocal Collision Avoidance

The original definition of the ORCA algorithm is considering the moving agents as *holonomic* point masses that do not have any constraints on the affecting accelerations, and thus can move in any direction at any time. This makes the algorithm very effective in guaranteeing collision-free paths. For this thesis the agents will not be considered as point masses, but instead as vehicles with a specified vehicle model having *non-holonomic* constraints. This will enforce several more constraints that the ORCA algorithm has to include in its calculations. The ORCA algorithm is therefore extended with a vehicle model, in this case the *Kinematic Bicycle model*, and the appropriate control structure to be able to include the behavior of vehicles in the collision avoidance calculations. This extension to the ORCA algorithm is proposed by [Alonso-Mora et al., 2012] and will from now on be called *Bicycle Optimal Reciprocal Collision Avoidance*, or in short *B-ORCA*.

Vehicle model

For this thesis the agents will be considered as non-holonomic vehicles that should behave as cars. The vehicle model used for this is the *Kinematic Bicycle model* [Kong et al., 2015] due to its simplicity and accurate behavior. The nonlinear equations for the models are as follows:

$$\dot{x} = v\cos(\Psi + \beta) \tag{2.11a}$$

$$\dot{y} = v\sin(\Psi + \beta) \tag{2.11b}$$

$$\dot{\Psi} = \frac{v}{l_r}\sin(\beta) \tag{2.11c}$$

$$\dot{v} = a \tag{2.11d}$$

$$\beta = \tan^{-1}\left(\frac{l_r}{l_f + l_r}\tan(\delta_f)\right). \tag{2.11e}$$

In the system of Equations (2.11), x and y are the coordinates for the vehicles center of mass in a global coordinate system (X,Y). The scalar v is the forward longitudinal speed of the vehicle and should not be confused with the velocity vector $\mathbf{v}$ from the center of mass of the vehicle. The angle Ψ represents the rotational position of the vehicle in the coordinate system (X,Y), and β the angle between the direction of the current velocity $\mathbf{v}$ and the longitudinal axis of the vehicle. The angle β is directly dependent on the steering angle δ_f and the distances l_f and l_r from the center of mass to the front and rear axes, respectively. The input parameters (control signals) to the system are the longitudinal acceleration a and the steering angle δ_f, see Figure 2.5 for a visualization of the model.

As changing the direction of the front wheel can not be done instantly, it is reasonable to also include a model for the steering. In this case, the steering angle

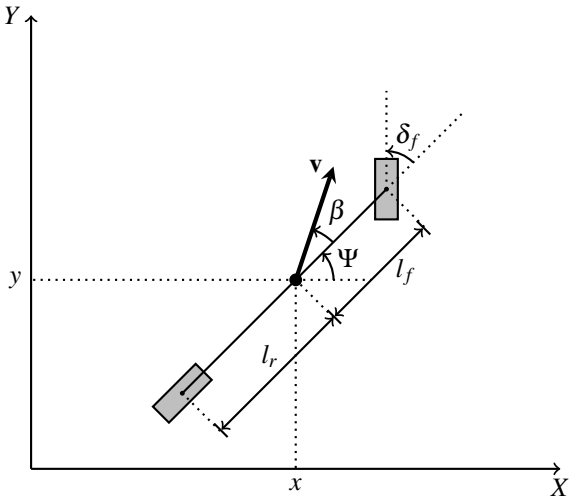

Figure 2.5 The Kinematic Bicycle model.

δ_f will be modeled as an integrator:

$$\dot{\delta}_f = \omega \tag{2.12}$$

where ω is the angular velocity of the steering wheel.

Discretized vehicle model To be able to use the *Kinematic vehicle model* in the ORCA algorithm it has to be discretized. This is done by applying the Euler method [Kong et al., 2015] which yield the formulation of the discretized system in 2.13, where $x(k+1)$ with a slight abuse of notation represents the discretization of the continuous time signal $x(t+\Delta t)$.

$$x(t+1) = x(t) + v\cos(\Psi(t)+\beta(t))\cdot\Delta t \tag{2.13a}$$
$$y(t+1) = y(t) + v\sin(\Psi(t)+\beta(t))\cdot\Delta t \tag{2.13b}$$
$$\Psi(t+1) = \Psi(t) + \frac{v}{l_r}\sin(\beta(t))\cdot\Delta t \tag{2.13c}$$
$$v(t+1) = v(t) + a(t)\cdot\Delta t \tag{2.13d}$$
$$\delta_f(t+1) = \delta_f(t) + \omega(t)\cdot\Delta t \tag{2.13e}$$
$$\beta(t) = \tan^{-1}\left(\frac{l_r}{l_f+l_r}\tan(\delta_f(t))\right) \tag{2.13f}$$

The input signals to the system in 2.13 are $a(t)$ and $\omega(t)$.

Control

The ORCA algorithm calculates a velocity $\mathbf{v}_A^{new}$ at every time step that should result in a collision-free trajectory within the time span τ. The vehicle should therefore ideally adopt $\mathbf{v}_A^{new}$, but as the vehicle model has *non-holonomic* constraints this will most often not be possible. The vehicle will instead do its *best* to follow the trajectory extrapolated from the velocity vector $\mathbf{v}_A^{new}$. The trajectory generated will be followed using a variation of a *pure pursuit controller* [Campbell, 2007], which uses a look-ahead point L to efficiently follow a desired straight trajectory. Figures 2.6 and 2.7 present how the pure pursuit control uses the look-ahead point to follow a straight trajectory. The distance error d_e, defined by the shortest distance between the look-ahead point and the desired trajectory, will be fed to a *PD*-controller to calculate the correct control signals to make the vehicle follow the trajectory.

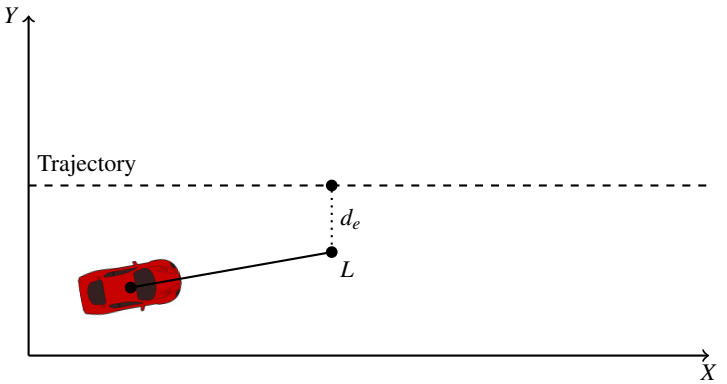

Figure 2.6 The look-ahead point L and the error d_e for a vehicle trying to follow a trajectory with pure pursuit control. When L is below the trajectory, as it is in this case, the controller will turn the front wheels to the left to move closer to the trajectory.

Steer angle control As mentioned above, the steering angle of the vehicle will be controlled with a PD-controller to implement pure pursuit of a trajectory. The input to the steering system is though the angular velocity of the steering angle, and not the steering angle itself which the PD-controller uses. To realize a controller for this problem, a cascaded controller is implemented, see Figure 2.8.

The subsystems $P_1(s)$ and P_2 of the control scheme in Figure 2.8 are created to be able to use a cascaded control scheme for the steering control. $P_1(s)$ is based on Equation (2.12) with the added tuning parameter k_s, and P_2 is based on (2.11). As

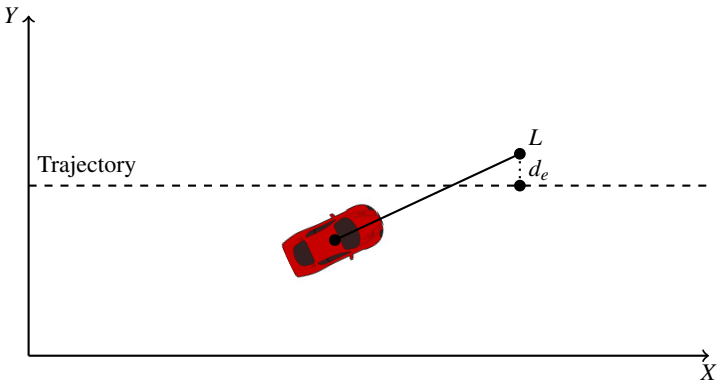

Figure 2.7 The look-ahead point L and the error d_e for a vehicle trying to follow a trajectory with pure pursuit control. When L is above the trajectory and the vehicle is below, as they are in this case, the controller will turn the front wheels to the right to compensate for overshoot.

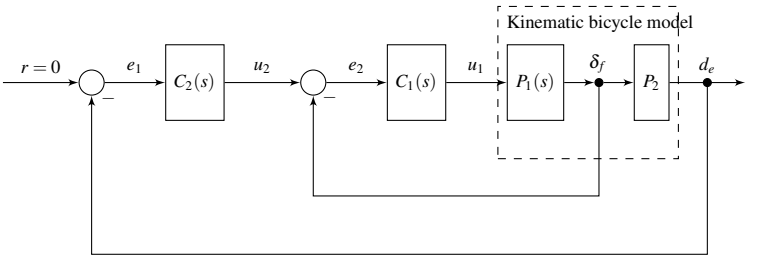

Figure 2.8 Cascaded control scheme for pure pursuit tracking of a straight line. The variable to control is the distance error d_e of the look-ahead point of a kinematic bicycle model. The kinematic bicycle model is divided into two subsystems, $P_1(s)$ and P_2, to be able to extract the steering angle δ_f of the vehicle which will be fed back to the controller. The input signal to the system is the angular velocity ω_f of the steer angle.

$P_1(s)$ is an integrator, a P-controller is used as the inner controller $C_1(s)$:

$$P_1(s) = \frac{k_s}{s} \tag{2.14}$$

$$C_1(s) = K_p. \tag{2.15}$$

The closed loop system for the inner loop is therefore:

$$P_{cl,inner} = \frac{P_1(s)C_1(s)}{1+P_1(s)C_1(s)} \tag{2.16}$$

$$P_{cl,inner} = \frac{k_s K_p}{s+k_s K_p}. \tag{2.17}$$

P_2 is a non-linear system and more complex than $P_1(s)$, so it can not be described with a transfer function. But as the pure pursuit of a trajectory can be done with a PD-controller [Campbell, 2007], which is the choice for $C_2(s)$:

$$C_2(s) = K_p + K_d s. \tag{2.18}$$

The control scheme presented above is in continuous time, but as the ORCA algorithm runs in discrete time the control scheme has to be discretized. A PID-controller can be discretized by first dividing the control in three parts than can be discretized separately [Åström et al., 2002]:

$$u(k) = u_P(k) + u_I(k) + u_D(k) \tag{2.19}$$

$u_P(k)$ in (2.21) follows directly from the continuous equivalent (2.20).

$$u_P(s) = K_p e \tag{2.20}$$

$$u_P(k) = K_p e(k) \tag{2.21}$$

$u_D(k)$ in (2.26) is derived by performing a backwards difference of the continuous equivalent (2.22) of the D-part with a low pass filter and the sampling time h.

$$u_D(s) = \frac{sT_D}{1+sT_D/N}(-Y(s)) \tag{2.22}$$

$$u_D(k) = \frac{T_D}{1+T_D h}u_D(k-1) - \frac{T_D N}{T_D+Nh}(y(k)-y(k-1)) \tag{2.23}$$

$$y(k) = d_e(k) \tag{2.24}$$

As the controller does not have an integral part, $u_I(k)$ is omitted. The full controller for the steering angle is therefore:

$$u_1(k) = K_{p,inner} e(k) \tag{2.25}$$

$$u_2(k) = K_{p,outer} e(k) + \frac{T_D}{1+T_D h}u_D(k-1) - \frac{T_D N}{T_D+Nh}(y(k)-y(k-1)) \tag{2.26}$$

Speed control Controlling the longitudinal speed v of the vehicle is more straight forward than the steering angle. The speed is modeled as the integral of the acceleration (2.11d), which is efficiently controlled by a P-controller. The reference speed for the vehicle to follow is $\|\mathbf{v}_A^{new}\|$.

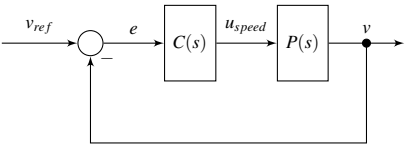

Figure 2.9 Control scheme for longitudinal speed control for a vehicle.

Figure 2.9 visualizes the control scheme for the speed control with the transfer functions

$$P(s) = \frac{1}{s} \tag{2.27}$$

$$C(s) = K_p. \tag{2.28}$$

The closed-loop system then becomes

$$P_{cl,speed} = \frac{P(s)C(s)}{1 + P(s)C(s)} \tag{2.29}$$

$$P_{cl,speed} = \frac{K_p}{s + K_p}. \tag{2.30}$$

Similar to (2.21), the discrete controller for speed control then becomes

$$u_{speed}(k) = K_p e(k). \tag{2.31}$$

Trajectory tracking Using the above proposed controllers to follow the desired velocity $\mathbf{v}_A^{new}$ can then be done by extrapolating this velocity vector to a straight trajectory, which the pure pursuit controllers can follow. The extrapolation of $\mathbf{v}_A^{new}$ will make the trajectory always go through the center of mass of the kinematic bicycle model, see Figure 2.10 for the geometry of the tracking problem. This makes the trajectory change for every time step, but this has no practical significance as the ORCA algorithm uses the safe velocities generated at every time step to create the collision-free trajectories.

Error bound

As the vehicle is subject to the non-holonomic constraints inherent to the Kinematic bicycle model, the safe new velocity $\mathbf{v}_A^{new}$ can not always be followed. This could become a problem when the vehicle does not follow the safe velocities calculated by the ORCA algorithm and therefore might crash onto another agent or a static obstacle. One solution to this problem is to specify an error bound to the total safety radius of the agents. This error bound ε can be selected based on the maximum tracking error the vehicle has by following different desired velocities $\mathbf{v}_d$. By pre-calculating the maximum tracking error for the kinematic vehicle model and the

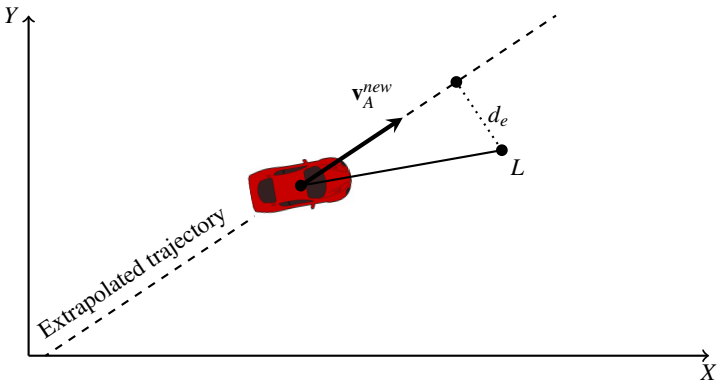

Figure 2.10 Tracking of the extrapolated trajectory of $\mathbf{v}_A^{new}$ by pure pursuit control. The distance error d_e is fed to the steering controller, and the speed $\|\mathbf{v}_A^{new}\|$ is fed to the speed controller.

controllers defined by (2.26, 2.31), it is possible to determine the set of desired velocities $\mathbf{v}_d$ that gives a tracking error smaller than ε_A for vehicle A, see Figure 2.11.

The possible velocities for agent A with a tracking error smaller than ε_A can formally be written as

$$\mathscr{V}_{\mathbf{v}_0,\Psi,\varepsilon} = \{\mathbf{v}_d | \mathscr{E}_{\mathbf{v}_0,\Psi}(\mathbf{v}_d) \leq \varepsilon\}. \tag{2.32}$$

The value ε_A is selected arbitrarily, but should not be too small as the set of viable velocities $\mathscr{V}_{\mathbf{v}_0,\Psi,\varepsilon}$ then will be very small. ε_A is therefore a tuning parameter and has to be tested for a good value.

The set $\mathscr{V}_{\mathbf{v}_0,\Psi,\varepsilon}$ is furthermore restricting which velocities that can be adopted by the vehicle in order to stay inside the error bound ε_A. The constraints used in the convex optimization calculation for finding the safe velocity $\mathbf{v}_A^{new}$ closest to $\mathbf{v}_A^{opt}$ will therefore be the intersection of $ORCA_A^\tau$ and $\mathscr{V}_{\mathbf{v}_0,\Psi,\varepsilon}$ as:

$$\mathbf{v}_A^{new} \in ORCA_A^\tau \cap \mathscr{V}_{\mathbf{v}_0,\Psi,\varepsilon}. \tag{2.33}$$

The error bound ε_A can be seen as the extension of the safety zone defined by r_A. To take into account the error bound for the collision avoidance, the sum $r_A + \varepsilon_A$ is used for the calculations of the Velocity Obstacles in (2.1). See Figure 2.12 for visualization of the geometry of the error bound.

Selection of possible velocities As the set $\mathscr{V}_{\mathbf{v}_0,\Psi,\varepsilon}$ defined in (2.32) is a complicated set, it is easier to approximate it by linear constraints. The best way to do this is to approximate it with a polyhedron with linear edges, as it makes solving the convex optimization (2.5) problem easier. If the set is approximated correctly, it is

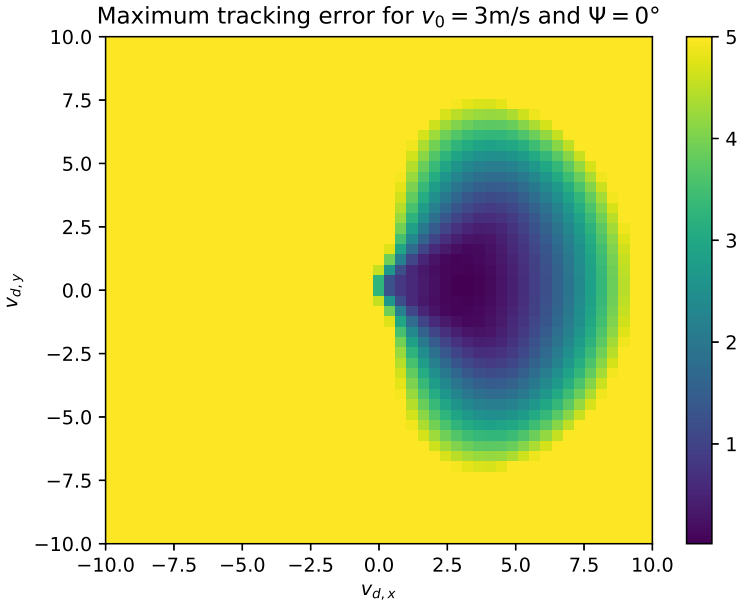

Figure 2.11 Maximum tracking errors $\mathscr{E}_{\mathbf{v}_0,\Psi}$ for different desired velocities $\mathbf{v}_d$, saturated at 5 m, with the initial states $\mathbf{v}_0 = 3\text{m/s}$ and $\Psi = 0°$.

possible to include the constraints in the convex optimization for efficient calculation of $\mathbf{v}_A^{new}$.

Collision Avoidance for non-holonomic agents

The total procedure for collision avoidance for the non-holonomic agent i can be summarized in the steps below. This is then done for every other agent to be controlled.

1. The preferred velocity $\mathbf{v}_i^{opt}$ is obtained. It is possible to select this in several ways, but to make the vehicles stop, $\mathbf{v}_i^{opt}$ is selected as the current velocity of the vehicle with a decreased absolute value of $a_{max} * \Delta t$ from the previous iteration.

2. The extended radius is selected as $r_i + \varepsilon_i$. If the agents are inside each others extended safety radii, it is instead selected as $r_i + \min(\varepsilon_i, (d(i,j) - r_i - r_j)/2)$. Here $d(i,j)$ is the distance between the middle points of agent i and agent j.

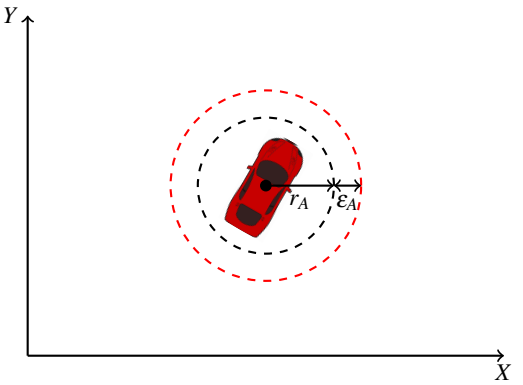

Figure 2.12 The vehicle with the safety zone defined by the circle of radius r_A, and the error bound defined by ε_A. The extended radius of the agent is therefore $r_A + \varepsilon_A$

3. The agents are considered as holonomic agents with the radius calculated in the previous step. From the original definition of the ORCA algorithm, the set of safe velocities $ORCA_i^\tau$ is calculated.

4. The new desired velocity $\mathbf{v}_i^{new}$ is calculated by solving the following convex optimization problem:

$$\mathbf{v}_i^{new} = \underset{\mathbf{v}_i \in ORCA_A^\tau \cap \mathscr{V}_{\mathbf{v}_0,\Psi,\varepsilon}}{\arg\min} \left(\|\mathbf{v}_i - \mathbf{v}_i^{opt}\| \right) \tag{2.34}$$

5. The trajectory extrapolated from $\mathbf{v}_i^{new}$ is tracked by the controllers defined by (2.26, 2.31) and the agent is moved to the new location which the discrete kinematic vehicle model (2.13) gets by updating its states according to the control signals.

Densely Packed Conditions

Rather than only using soft constraints as done in the original formulation of the ORCA algorithm presented in Section 2.1 when there is no viable set of safe velocities because $ORCA_A^\tau = \emptyset$, this formulation of the B-ORCA decreases the time horizon τ in an iterative way. The time horizon τ_{dyn} for dynamic obstacles and τ_{static} is set to a nominal value, as they are tuning parameters of the algorithm. If there are no viable safe velocities because $ORCA_A^\tau \cap \mathscr{V}_{\mathbf{v}_0,\Psi,\varepsilon} = \emptyset$, the time horizon for collision is decreased by half, $\tau_{dyn} = \tau_{dyn}/2$ and $\tau_{static} = \tau_{static}/2$, and the set $ORCA_A^\tau$ is calculated again. This is done until a set of viable velocities is found, or $\tau < \tau_{min}$, where τ_{min} is a minimum value for the time horizon and a tuning parameter. If this threshold is reached, soft constraints will be applied to avoid collision. Following this procedure makes the vehicle prioritize the closest obstacles to ensure collision-free trajectories.

3

Implementation

In this chapter a detailed explanation of the implementation of the ORCA algorithm is presented.

3.1 Programming framework

The implementation of the *Bicycle reciprocal collision avoidance* algorithm for this thesis is done in the programming language *Python 3.6*. Python is a very powerful programming language often used in scientific programming and software development. Due to Python's easy syntax and structure it goes relatively fast to develop advanced algorithms. As Python is widely used and open source, it is possible to use libraries from the community for almost any task.

Python can be summarized as an *Interpreted*, *Object-Oriented*, *High-Level* programming language. An interpreted programming language means that it is not required to compile the code in order be able to run the program, which greatly speeds up the development cycle. The high abstraction level also reduces the development time, as the programmer does not have to think about the details of the computer when developing. This also makes the code more much readable and easier to maintain than low-level programming languages. Furthermore, due to Python's Object-Oriented structure it is easy to write modular code that is easy to reuse. This speeds up the development process and allows easy use of external libraries.

Python is thus a very effective language when prototyping and developing algorithms. These positive aspects though comes with a downside; the run time on the computer is much slower than more low-level programming languages. For this project it was decided to prove the concept of the B-ORCA algorithm with Python as the development time of implementing the algorithm in C would be overwhelming. Python also comes with libraries that allow to port Python code into C code for faster run times on the computer [Behnel et al., 2019].

Libraries

For the implementation of the B-ORCA algorithm, the majority of the external libraries used were installed with the *Anaconda Package Manager*. This package manager is widely used for scientific use of Python and when installing Anaconda it already comes with the most popular scientific libraries already installed.

Below follows the three most prominent external libraries used for the development of the B-ORCA algorithm.

NumPy *NumPy* is a library in Python for numerical operations with very powerful and fast matrix operations. It is the base library of which many other scientific libraries build upon and besides the fast matrix operations it has powerful capabilities of random number generation, Fourier transform and linear algebra. The *NumPy* library comes with the installation of the Anaconda package manager.

Matplotlib *Matplotlib* is the most popular library for visualization of data in Python. With this library it is possible to plot and visualise very complex data as easily as possible, and it has functions for all relevant plot types and simple animations. The *Matplotlib* library comes with the installation of the Anaconda package manager.

Quadprog *Quadprog* is a library for solving convex optimazation problems as stated in (2.5). The routine for solving the convex problem is the Goldfarb/Idnani dual algorithm [Goldfarb and Idnani, 1982]. This library is not included in the Anaconda package manager and has to be installed seperately. The installation of the Quadprog library can be done with the *pip package manager*.

3.2 Algorithm description

The B-ORCA algorithm was implemented in Python according to the theory presented in Chapter 2 and the resulting algorithm can be summarized in Algorithm 1 described below. A detailed description for every code section is given below.

- *Preliminaries*: The input to the algorithm are the current states of the test scenario. These states are the *(X,Y)-location*, *velocity* and *heading angle* for every vehicle in the scenario, and also the locations of the static obstacles. The output from the B-ORCA algorithm are the safe trajectories for every vehicle in the scenario. The trajectories consist of trajectory nodes that contain the *time*, *(X,Y)-location*, *velocity* and *heading angle* of the vehicle.

- *Lines 2-4*: The algorithm begins with creating an empty collection of future trajectories T. There will be one trajectory for every vehicle and for every iteration of the algorithm, one new trajectory node will be appended to every trajectory. The dynamic A and static O elements are then extracted from the

scenario S to be able to perform operations on them. The data of the dynamic elements A is now corresponding to the initial states of the vehicles.

- *Line 5*: The algorithm will continue until all the vehicles have stopped.
- *Line 6*: Create a set of empty velocities that will be applied to the vehicle agents. This list will consist of one new velocity vector for every vehicle.
- *Line 8*: Loop through and calculate the safe velocity for every vehicle agent a in A.
- *Line 9*: Create the empty set of the safe velocities the vehicle agent a can choose among to not collide with any object or obstacle.
- *Line 10*: Calculate safe velocities as long as the set $ORCA_a$ is empty.
- *Line 11*: Create the set $ORCA_a^\tau$ that consists of the continuous space $\mathbb{R}^2$ with the prediction time horizon τ. By the following lines (12-21) this set will be shrunk to only consist of the safe velocities for a.
- *Lines 12-17*: Calculate the safe velocities $ORCA_{a|b}^\tau$ for agent a to not collide into vehicle agent b. This calculation is described by (2.4). This constraint is then added to $ORCA_a^\tau$ for every agent b to limit the set of safe velocities.
- *Lines 18-21*: Calculate the safe velocities $ORCA_{a|o}^\tau$ for agent a to not collide into the static obstacle o. This calculation is described by (2.9). This constraint is then added to $ORCA_a^\tau$ for every obstacle o to limit the set of safe velocities.
- *Lines 21-22*: Assign the calculated set with the safe velocities for agent a $ORCA_a^\tau$ to the empty set $ORCA_a$. There is though still a possibility that the set $ORCA_a$ is still empty. In that case the procedure of the while loop stated at line 10 is repeated but with the prediction horizon divided by half; $\tau = \tau/2$. Decreasing the prediction horizon will therefore prioritize the closest obstacles to vehicle agent a.
- *Lines 25-26*: Calculate the new velocity $\mathbf{v}_a^{new} \in ORCA_a$ for agent a which is closest to the desired velocity $\mathbf{v}_a^{opt}$ and store it in V^{new}.
- *Lines 29-30*: When the new velocities of all vehicle agents in A are calculated they are applied to update the states of the agents in A. The states of the agents are updated based on the discrete-time equations for the kinematic bicycle model (2.13) and the path following controller defined by (2.25), (2.26) and (2.31). The updated states of every agent are then saved as a trajectory node in their corresponding trajectories in T.

Algorithm 1: B-ORCA

Input: S, The current states of the scenario
Output: T, The safe trajectories

begin
  $T = \emptyset$
  $A = ExtractDynamicElements(S)$
  $O = ExtractStaticElements(S)$
  while *Agents still moving* **do**
    $V^{new} = \emptyset$
    // Calculate safe velocities
    for $a \in A$ **do**
      $ORCA_a = \emptyset$
      while $ORCA_a = \emptyset$ ***and*** $\tau > \tau_{min}$ **do**
        $ORCA^{\tau}_a = \mathbb{R}^2$
        for $b \in A$ **do**
          if $a \neq b$ **then**
            $ORCA^{\tau}_{a|b} = CalculateSafeVelocities(a,b)$
            $ORCA^{\tau}_a = ORCA^{\tau}_a \cap ORCA^{\tau}_{a|b}$
          end
        end
        for $o \in O$ **do**
          $ORCA^{\tau}_{a|o} = CalculateSafeVelocities(a,o)$
          $ORCA^{\tau}_a = ORCA^{\tau}_a \cap ORCA^{\tau}_{a|o}$
        end
        $ORCA_a = ORCA^{\tau}_a$
        $\tau = \tau/2$
      end
      $\mathbf{v}^{new}_a = ConvexOptimization(\mathbf{v}^{opt}_a, ORCA_a)$
      $V^{new}.append(\mathbf{v}^{new}_a)$
    end
    // Apply new velocities and update agents' positions
    $A = UpdateAgentsStates(A, V^{new})$
    $T.append(A)$
  end
end

4

Experimental Procedure

4.1 Experimental overview

Due to the complexity of the B-ORCA algorithm, it is hard to perform an analytical evaluation of its performance. To be able to analyze the algorithm's efficiency an empirical study is done instead. The study of the algorithm is made by performing experiments over a certain set of scenarios that are typical when evaluating autonomous driving. The scenarios are designed to be simple in their format, but still capture the performance of the algorithm tested. The scenarios selected for the experiments are *Wall collision*, *Head on collision*, *Collision from angle*, and *Overtake*, which are presented in the next section.

The goal of the empirical study was to push the B-ORCA algorithm to its limits to see when it succeeds and when it fails to find safe trajectories. In the first step, a large set of trajectories was generated, one for every configuration of every scenario. After that, a selection of the generated trajectories was further evaluated to guarantee their quality, in the sense of how easy it is for a vehicle to follow the trajectories. This was done by feeding the trajectories to the simulation module SPAS, the Driving robot simulator and the real Driving robot, which are described in this chapter. The data is extracted and analyzed to see if the simulated and real vehicles stay inside the safety zones, visualized in Figure 2.12, of the B-ORCA algorithm, which are collision-free if the algorithm succeeds. A flowchart of the experimental procedure is presented in Figure 4.1.

Simulation Platform for Active Safety (SPAS)

Simulation Platform for Active Safety (SPAS) is a MATLAB/Simulink based simulation platform developed at Volvo Car Corporation. SPAS enables virtual testing and verification of vehicles, active safety systems and autonomous drive systems in different specified traffic environments. To do this, SPAS models the driver, power plant, transmission, drive-line, chassis, brakes, steering etc. which together build the full vehicle model. SPAS also takes into consideration sensors and environmental objects like roads, road signs, pedestrians and other vehicles.

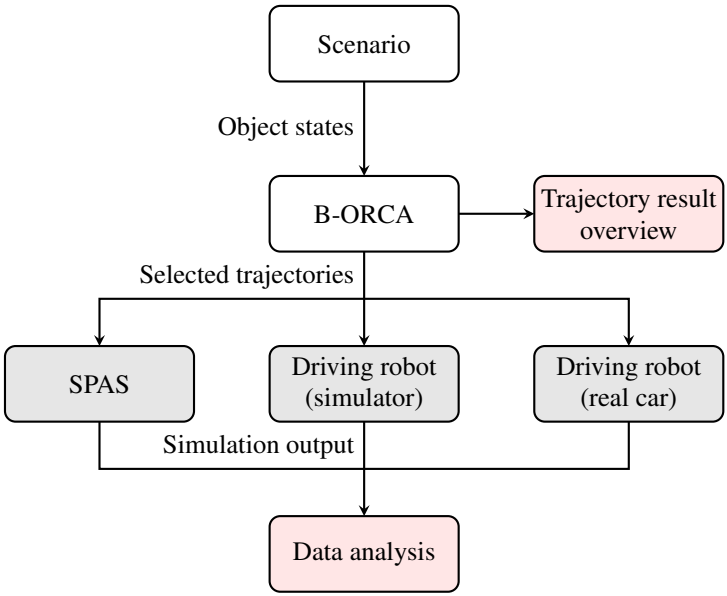

Figure 4.1 Flowchart of the experimental procedure of a scenario. The *B-ORCA* algorithm gets the states of the objects of the scenarios and calculates the safe trajectories for each configuration. When all the configurations of the scenario has been run, the result is presented in the *Trajectory result overview*. A smaller set of trajectories are selected for further analysis. This is done by passing the trajectories through *SPAS*, the *Driving robot simulator*, and the *Driving robot*. The result of these simulations are then saved for further analysis to evaluate the trajectories generated by *B-ORCA*.

Not all functionality of SPAS was used to evaluate the trajectories generated from the B-ORCA algorithm. The simulations were set up so that the selected trajectories were fed to SPAS one by one. SPAS then created a custom road and initialized the car in the beginning of this road with the correct initial speed. As the simulation starts, the car follows the road as accurately as possible with the desired speed profile set as speed reference. When the simulation ends, the output data can be acquired for further analysis. The data of interest for evaluating the performance of the B-ORCA algorithm are mainly:

- time series of (X,Y) location,
- cross-track error,
- distance to desired trajectory node,
- velocity profile,

- longitudinal and lateral acceleration.

Driving robots

The Driving robots are created by a company that develops and delivers various testing equipment for the automotive market. Their product range are mostly focused on vehicle testing on the test track with robots for vehicle control and moving obstacles for testing ADAS systems. The company also provides a common software interface which makes it easy to use and connect many objects for collaboration. For this thesis, two different Driving robots will work together to control each vehicle that was tested. These robots are the *Steering robot* and *Pedal robot*. The company also delivers a robot simulator which makes it possible to run simulations of the behavior of the robots before testing them on the test track.

Steering robot To be able to steer the car in a desired direction the steering robot applies the appropriate torque to the steering wheel. The robot is mounted on the steering wheel and is also attached to the frame of the car. The actuator is a motor that is located in the robot and which turns the steering wheel with the desired torque. How the steering robot is mounted in the car is shown in Figure 4.2.

Pedal robot The pedal robot takes control of the braking and the acceleration of the car to control the speed. The robot is mounted in the leg space in front of the driver seat and is actuated by two motors, one for each pedal in a car with automatic transmission. How the pedal robot is mounted in the car is shown in Figure 4.2.

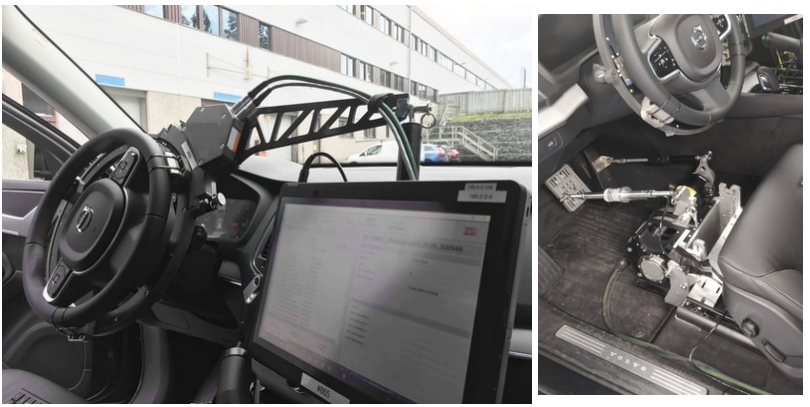

Figure 4.2 **(Left)** The steering robot used in the experiments. **(Right)** The pedal robot used in the experiments.

Path following Combining the Steering and Pedal robot makes it is possible to take full control of the car. By also including a *Motion pack* that tracks the car's exact position with *GPS*-signals and an *Inertial Measurement Unit*. With knowledge

of the current position, velocity and heading of the car, it will then be possible for the robots to control the car so that it follows a desired path. Figure 4.3 shows the block diagram of the Driving robot controller for path following.

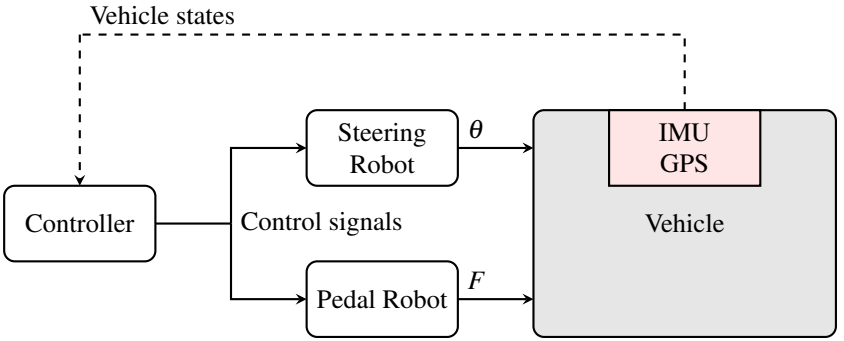

Figure 4.3 Block diagram of the driving robot controller for path following. The controller sends control signals to the steering and pedal robot so that they can actuate the correct steering wheel angle θ and pedal force F to control the car. The states of the car are then measured by the *IMU* and the *GPS* and are fed back to the controller.

Robot simulator To prepare and plan tests before running them on the real Driving robots it is possible to use a robot simulator. This simulator has the exact same software interface as the real Driving robots but all the robots and the car itself are simulated. Paths that the car should follow can therefore be evaluated for their viability in real-time before actually running them on the real system. Figure 4.4 shows which parts of the original path following structure that are simulated. The setup used for the simulations is shown in Figure 4.5.

4.2 Scenarios

To evaluate the performance of the B-ORCA algorithm, four scenarios are set up. These scenarios are *Wall collision*, *Head on collision*, *Collision from angle*, and *Overtake*, and are presented further below. Every scenario will have a set of parameters associated to the initial position and velocity of the vehicles which are possible to adjust to be able to test the algorithm for different configurations of the scenario. For every scenario two parameters were changed to adjust the configuration. In the cases when the scenario has more parameters, the two most relevant were selected. With two variables that are adjusted to change the configuration of the scenario it is possible to draw a 2-D map of when the algorithm succeeds or fails, which will give a describing overview of the results.

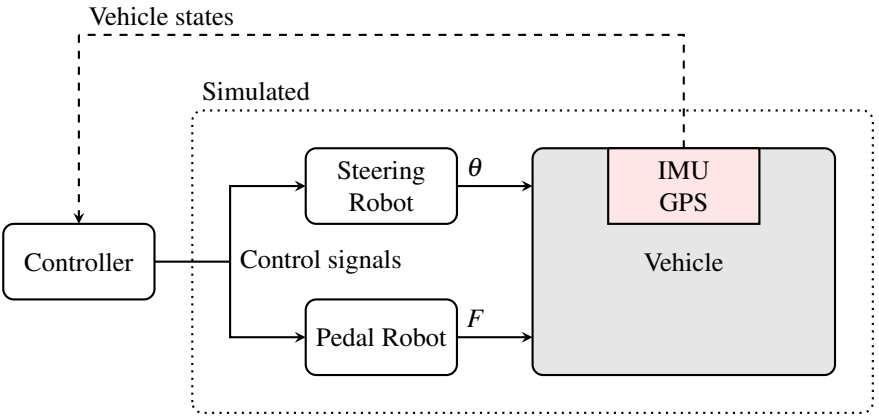

Figure 4.4 Block diagram of the driving robot controller for simulation of path following. This is done in the same way as the regular path following, but the Steering robot, the Pedal robot and the Vehicle are simulated.

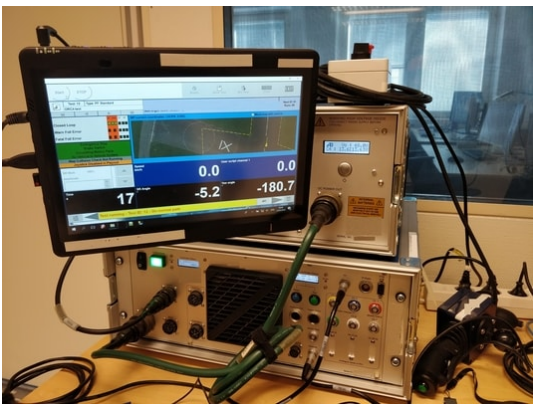

Figure 4.5 Setup of driving robot simulator.

For all the scenarios and their configurations, the parameters of the B-ORCA algorithm were the same to give a fair comparison of its performance. It is possible to optimize the parameters of the algorithm for every scenario to get the optimal performance, but that is not done in this thesis as the focus is more to give an overview of its performance for different scenarios. The parameter configuration of the B-ORCA algorithm used in the experiments for the different scenarios are presented in Table 4.1.

Table 4.1 Parameter configuration of the B-ORCA algorithm for the scenarios used in the evaluation of the algorithms performance.

Parameter description	Parameter	Value	Unit
Safety boundary radius	r	3	m
Error boundary radius	ε	0.5	m
Max longitudinal acceleration	$a_{long,max}$	4	m/s^2
Max front wheel turn angle	δ_f	0.2	rad
Static obstacle collision time horizon	τ_{static}	20	s
Dynamic obstacle collision time horizon	τ_{dyn}	20	s
Center of mass to front axle distance	l_f	1.5	m
Center of mass to rear axle distance	l_r	1.5	m
Vehicle length	l_{veh}	4.9	m
Vehicle width	w_{veh}	1.9	m

Wall collision

The *Wall collision* scenario is a basic scenario with one car driving perpendicular towards a wall. See Figure 4.6 for a bird's view of the scenario. The scenario has two configurable parameters, $\mathbf{v}_1$ and d_1, which change the layout of the scenario. The input parameters when performing the experiments are the speed v of the vehicle directed towards the wall, and the distance to the wall d. Many configurations

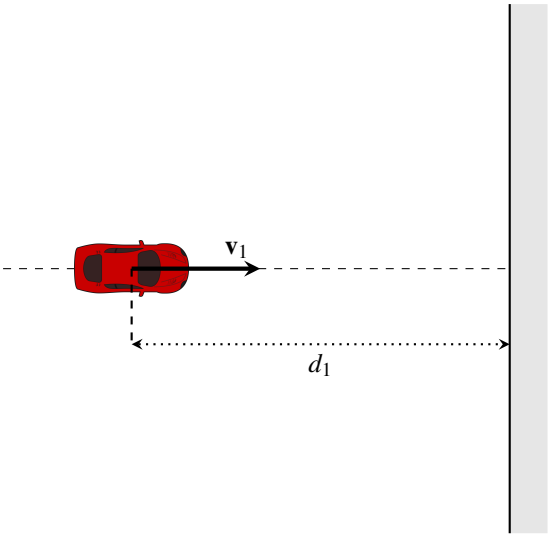

Figure 4.6 Bird's view of the Wall collision scenario. The car is driving straight towards a wall with velocity $\mathbf{v}_1$ with the distance d_1 from the wall.

Table 4.2 Parameter configuration of the *Wall collision* scenario. Every input parameter has a range of values that are used to modify the scenario in order to stress the B-ORCA algorithm. The input parameters are then mapped to the scenario parameters, which are also visualized in Figure 4.6.

Input parameters			
Parameter description	**Parameter**	**Values**	**Unit**
Vehicle speed	v	$[0, 30]$	m/s
Distance to wall	d	$[0, 30]$	m

Scenario parameters			
Parameter description	**Parameter**	**Value**	**Unit**
Vehicle speed	$\mathbf{v}_1$	v	m/s
Distance to wall	d_1	d	m

of the scenario were tested to stress the B-ORCA algorithm. How these input parameters are mapped to the scenario parameters shown in Figure 4.6 is presented in Table 4.2. In this scenario, the input parameters are mapped directly to the scenario parameters, which will not be the case for the other scenarios.

Head on collision

The *Head on collision* scenario is defined by two cars driving straight towards each other with a collision imminent in the near future. The cars have the velocities $\mathbf{v}_1$ and $\mathbf{v}_2$ and are separated with the distance d_1. See Figure 4.7 for a bird's view of the scenario. Many configurations of the scenario were tested to stress the B-ORCA algorithm. Two input parameters – v and d – were changed to modify the configuration of the scenario. How these input parameters are mapped to the scenario parameters shown in Figure 4.7 is presented in Table 4.3.

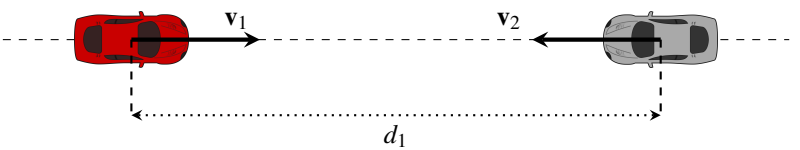

Figure 4.7 Bird's view of the Head on collision scenario. Two cars are driving straight towards each other with velocity $\mathbf{v}_1$ and $\mathbf{v}_2$. The cars are separated with the distance d_1.

Collision from angle

The *Collision from angle* scenario is defined by two cars that are driving straight but heading for a collision in the near future as their paths will intersect. The cars are driving with the velocities $\mathbf{v}_1$ respectively $\mathbf{v}_2$ towards the intersection point from

Table 4.3 Parameter configuration of the *Head on collision* scenario. Every input parameter has a range of values that are used to modify the scenario in order to stress the B-ORCA algorithm. The input parameters are then mapped to the scenario parameters, which are also visualized in Figure 4.7

Input parameters			
Parameter description	**Parameter**	**Values**	**Unit**
Vehicle speed	v	$[0, 30]$	m/s
Distance between vehicles	d	$[0, 30]$	m

Scenario parameters			
Parameter description	**Parameter**	**Value**	**Unit**
Vehicle 1 speed	$\mathbf{v}_1$	v	m/s
Vehicle 2 speed	$\mathbf{v}_2$	v	m/s
Distance between vehicles	d_1	d	m

which they are separated with the distance d_c when the safe trajectories are calculated. The collision angle of the cars is α_c. A bird's view of the scenario is presented in Figure 4.8. Many configurations of the scenario were tested to stress the B-ORCA algorithm. Two input parameters – v and d – were changed to modify the configuration of the scenario. How these input parameters are mapped to the scenario parameters shown in Figure 4.8 is presented in Table 4.4.

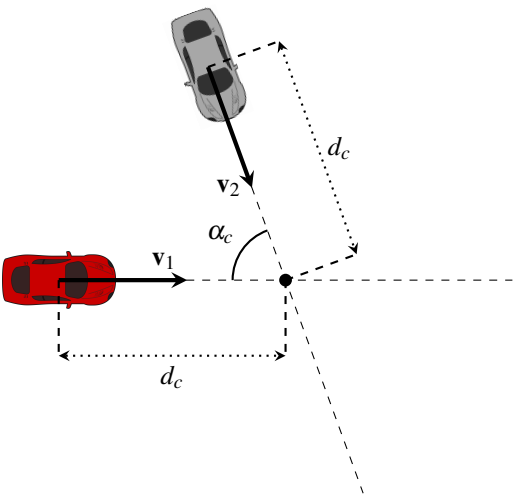

Figure 4.8 Bird's view of the *Collision from angle* scenario. Two cars are driving straight with the velocities $\mathbf{v}_1$ respectively $\mathbf{v}_2$ and will reach an intersection point where they will collide. They are separated from this intersection point with the distance d_c. The collision angle between the cars is α_c.

Table 4.4 Parameter configuration of the *Collision from angle* scenario. Every input parameter has a range of values that are used to modify the scenario in order to stress the B-ORCA algorithm. The input parameters are then mapped to the scenario parameters, which are also visualized in Figure 4.8

Input parameters			
Parameter description	**Parameter**	**Values**	**Unit**
Vehicle speed	v	$[0, 30]$	m/s
Distance to intersection	d	$[0, 30]$	m

Scenario parameters			
Parameter description	**Parameter**	**Value**	**Unit**
Vehicle 1 speed	$\mathbf{v}_1$	v	m/s
Vehicle 2 speed	$\mathbf{v}_2$	v	m/s
Distance to intersection	d_c	d	m
Collision angle	α_c	90	deg

Overtake

The *Overtake* scenario is more complex than the previous scenarios. It involves three cars driving on a straight road limited by two static obstacles on both sides, which represents the road boundaries. Vehicle 1 attempts to overtake vehicle 2, but during the overtake vehicle 3 is heading onto vehicle 1, see Figure 4.9. The velocities of the three cars are $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$ respectively. $\mathbf{v}_1$ is larger than the other two velocities as the overtaking car is driving faster than the others. The overtaking car is separated longitudinally from the second car with the distance d_1. The second car is then separated longitudinally from the third car with the distance d_2. The lateral separation of the two lanes is w_l, and the separation from the lanes to the walls on the sides is w_s. Many configurations of the scenario were tested to stress the B-ORCA algorithm. Two input parameters – v and d – were changed to modify the configuration of the scenario. How these input parameters are mapped to the scenario parameters shown in Figure 4.9 is presented in Table 4.5.

4.3 Trajectory evaluation

A selection of the trajectories generated by the B-ORCA algorithm is run through SPAS, the Driving robot simulator and the real Driving robots to evaluate the quality of the trajectories. The trajectory quality is measured in the sense of how easy it is for a vehicle to follow the desired trajectory, which amongst others will tell how accurate the vehicle model used to develop the B-ORCA algorithm is. The B-ORCA algorithm uses circular safety zones around the vehicle to guarantee collision free trajectories. It is therefore important that all vehicles stay inside their safety zones at all times when following a trajectory as the B-ORCA algorithm does not guarantee that no collision will occur if any vehicle drifts outside its dedicated safety zone. The

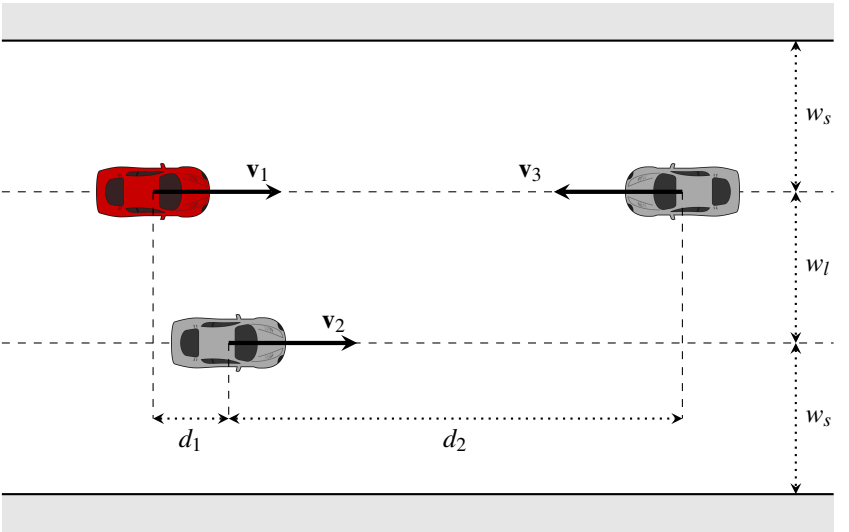

Figure 4.9 Bird's view of the *Overtake* scenario. This scenario consists of three cars driving on a straight road. The first cars tries to overtake the second car but suddenly sees an oncoming third car in the opposite lane. The velocities of the vehicles are $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$. The vehicles are longitudinally separated by the distances d_1 and d_2 and the lateral separation between the lanes is w_l. The distances of the center of the lanes to the closest wall is then w_s.

accelerations acting on the vehicle are also of interest as they couples to the vehicle model and also give an indication of the risk of the tests, as high acceleration and fast turning at high speed can be dangerous.

For the evaluation of the trajectories, seven measurements were inspected to get an image of how easy it is to follow the trajectories. These measurements are *Trajectory tracking*, *Velocity tracking*, *Velocity tracking error*, *Lateral acceleration*, *Longitudinal acceleration*, *Path following error* and *Positional displacement*. These measurements describes different aspects of trajectory following and together they provide sufficient information for evaluating the trajectory quality.

Trajectory tracking The trajectory tracking measurement is an overview of the tested scenario in a (x, y)-plane. The driven trajectories of all the vehicles in the scenario is presented and compared to their desired trajectories. This measurement is used to give an intuitive view of the test and see the vehicles relative positions to each other.

Lateral trajectory following error The Lateral trajectory following error is the vehicle's error perpendicular to the desired trajectory measured from its front axle. This measurement is an effective way of observing the lateral displacement of the

Table 4.5 Parameter configuration of the *Overtake* scenario. Every input parameter has a range of values that are used to modify the scenario in order to stress the B-ORCA algorithm. The input parameters are then mapped to the scenario parameters, which are also visualized in Figure 4.9.

Input parameters			
Parameter description	**Parameter**	**Values**	**Unit**
Vehicle speeds	v	$[0,30]$	m/s
Overtake distance	d	$[-20,20]$	m

Scenario parameters			
Parameter description	**Parameter**	**Value**	**Unit**
Vehicle 1 speed	$\mathbf{v}_1$	$v+5$	m/s
Vehicle 2 speed	$\mathbf{v}_2$	v	m/s
Vehicle 3 speed	$\mathbf{v}_3$	v	m/s
Distance, vehicle 1 to vehicle 2	d_1	d	m
Distance, vehicle 2 to vehicle 3	d_2	35	m
Lane separation	w_l	7	m
Wall separation	w_s	7	m

vehicle from its desired path without taking the timestamps of the trajectory nodes into consideration. See Figure 4.10 for a visual explanation.

Positional displacement The Positional displacement measurement is similar to the Lateral trajectory following error, but instead of taking the error perpendicular to the desired path, the error is measured from the current position of the front axle to the trajectory node with the timestamp that matches the current time. This measurement is effective as it quantifies how much the vehicle is deviating outside of its safety zone, see Figure 4.10 for a visual explanation.

Speed tracking The Speed tracking measurement shows how the desired velocity profiles are followed over time. This is useful because too large errors in the tracking will result in displacements of the vehicles relative to their desired locations at a given time.

Speed tracking error The speed tracking error measurement is based on the same information as the speed tracking measurement but presented in an alternative way. The definition of the measurement is $e_{speed} = v_{real} - v_{desired}$, and a perfect speed tracking will give the error measurement zero. The maximum error measurement will also be extracted for benchmarking.

Lateral acceleration The lateral acceleration measurement shows the acceleration of the vehicle sideways over time, as for example when the vehicle turns. This is an indication on how strong the forces acting on the vehicle are and how risky the trajectory is, since only certain levels of lateral acceleration can be achieved with a real car before it starts drifting or looses control of where it is going.

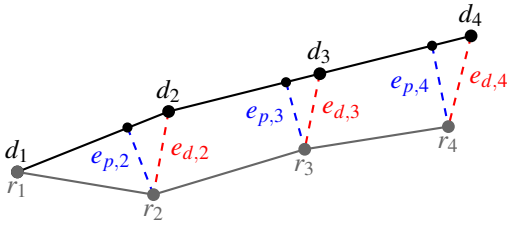

Figure 4.10 Visualisation of the *Lateral trajectory following error* and the *Positional displacement* measurements. The vector $\mathbf{d} = (d_1, d_2, d_3, d_4, ...)$ is the time series of the desired path and the vector $\mathbf{r} = (r_1, r_2, r_3, r_4, ...)$ is the time series of the resulting front axle path of the vehicle. The Positional displacement $\mathbf{e}_d$ is depending on the time stamp of the measurements and is calculated $e_{d,i} = r_i - d_i$. The Lateral trajectory following error $\mathbf{e}_p$ is instead calculated as the closest point to the desired path.

Longitudinal acceleration The longitudinal acceleration measurement shows the acceleration of the vehicle in the direction of travel over time. It gives a good indication on how hard the vehicle brakes and accelerates.

Safety constraints

The B-ORCA algorithm provides an output if the generated trajectories are safe, which in other words means that the vehicles safety zones are not intersecting with each other or the static obstacles. If the vehicles follows these trajectories perfectly it is known that the vehicles will not collide into each other. A problem arises when the vehicles do not follow the safe trajectory perfectly, due to factors as for example non-viable trajectories or poorly tuned controllers. If the vehicles deviate outside the zones that the B-ORCA algorithm guarantees to be safe, it is no longer possible to guarantee that collisions will not happen.

It is therefore important to analyse how well the vehicles follow the trajectories given to see when they deviate too much and start leaving the safety zone. The best measurement for this is to analyse the *Positional displacement* and the *Lateral trajectory following error*. To say that the vehicle is guaranteed to be inside the safety zone the most conservative constraint is to take the absolute shortest distance from the vehicle to the safety zone, see measure $l_{closest}$ in Figure 4.11. This measure is therefore used as the safety constraint of the *Positional displacement* measured. As mentioned, this is a very conservative constraint and several examples were encountered in which this constraint is violated but the vehicle is still inside the safety zone, but to be able to guarantee that the vehicles does not exit its safety zones, the upper limit on positional displacement has to be this conservative. An example of when $l_{closest}$ is violated but the vehicle is still inside the safety zone is when the vehicle is just displaced perpendicular to the trajectory to travel and the *Positional*

displacement (e_{disp}) is inside the bound $l_{closest} < e_{disp} < l_{perp}$, see Figure 4.11 for reference.

Considering the constraint for the *Lateral trajectory following error*, this has not to be as conservative as the measurement is not depending on the current time. Here the constraint is only the lateral displacement relative to the path to travel. See measure l_{perp} in Figure 4.11.

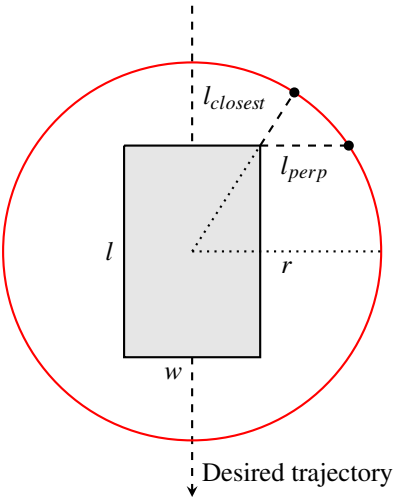

Figure 4.11 Safety distances from the vehicle to the edge of the safety zone. The distance $l_{closest}$ is the most conservative constraint on the path deviation. The distance l_{perp} is a more relaxed constraint as it only limits the perpendicular displacement of the vehicle.

From geometry the conservative constraint $l_{closest}$ is given by

$$l_{closest} = r - \sqrt{\left(\frac{w}{2}\right)^2 + \left(\frac{l}{2}\right)^2} \tag{4.1}$$

and the perpendicular constraint l_{perp} by

$$l_{perp} = r\cos\left(\arcsin\left(\frac{l}{2r}\right)\right) \tag{4.2}$$

With the set of parameters used for the experiments (specified in Table 4.1) the constraints are given by the values presented in Table 4.6.

Table 4.6 Values for displacement constraint to guarantee no vehicle collisions.

Parameter description	Parameter	Value	Unit
Absolute displacement constraint	$l_{closest}$	0.33	m
Perpendicular displacement constraint	l_{perp}	0.71	m

5

Results

This chapter presents the resulting performance of the B-ORCA algorithm. First, a map of the success will be given for every scenario defined in Section 4.2 to show when the B-ORCA algorithm succeeded in finding safe trajectories. After that, selected simulation results will be presented to showcase how well the vehicles follow the given trajectories. Only the *Overtake* scenario will be presented in this section to improve readability. The interested reader can find the results of the other scenarios in Appendix A.

An overview of the performance of the algorithm will then be given by combining all simulation results from the four different scenarios to broaden the perspective of the results. What is evident when running the simulations when it comes to the success of vehicles following the given trajectories is that the success rate is highly correlated to the maximum lateral acceleration of the vehicles, which in turn is highly correlated to the speed of the vehicles. The result will therefore be focused on the correlation between the vehicle' speeds, maximum lateral accelerations and the success of following the desired trajectories.

The results are mainly based on measurements from the driving robot simulator. From the simulator, around 100MB of data was extracted which was based on around 230 experiments of simulated vehicles attempting to follow trajectories generated by the B-ORCA algorithm. A selection of the tests performed on the driving robot simulator were also performed on a real car to get data from real test cases. From the real driving robot, 300MB of data was extracted which was based on 82 tests. The majority of the data were collected from the driving robot simulator so the real data is be used to evaluate how accurate the simulator is. Also, some SPAS simulations were conducted, but the speed following controller implemented in the simulated vehicles was not designed for this purpose and performed poorly, which disturbed the measurements. The result from the SPAS simulations were therefore excluded from this analysis to not mislead the reader.

A discussion of the result presented in this chapter will be covered in Chapter 6.

5.1 Overtake scenario simulations

This section focuses on presenting the results of when the B-ORCA algorithm generates trajectories to avoid collision between vehicles in the *Overtake* scenario presented in Section 4.2. A map over the successful trajectory generation will be presented to show in which configurations the B-ORCA algorithm succeeds in avoiding an imminent collision. After that, the simulation result of the Driving robot simulator trying to follow the calculated trajectories are presented. The configuration of the *overtake* scenario discussed here corresponds to the input parameters $v = 14$m/s and $d = 4$m (see Table 4.5 for definition), of which the B-ORCA algorithm succeeded in calculating safe trajectories according to Figure 5.1.

Success of trajectory generation

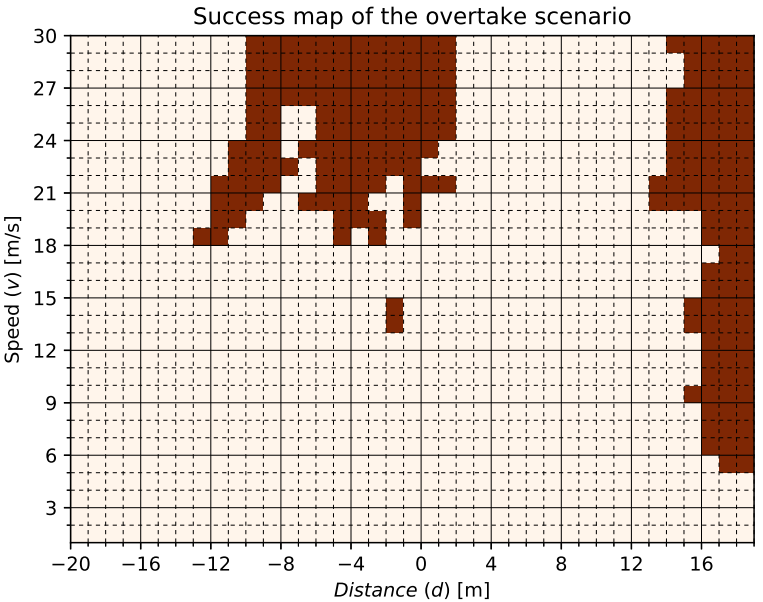

Figure 5.1 Success map of the path generation of the overtake scenario. Red color symbolizes an unsuccessful trajectory generation of the B-ORCA algorithm, and white color symbolizes a successful trajectory generation. The values on the axes are the input parameters of the scenario, defined in Table 4.5. An exemplification of the success map can be found in Appendix B.

Manual analysis of trajectory viability Figure 5.1 shows the success map of the overtake scenario trajectory generation, where a successful trajectory generation is defined by no intersection of the vehicles' safety zones. For every square of a (v,d) value pair in Figure 5.1, it is also possible to manually analyze the generated trajectories. This is done in Figure 5.2. Observe that the safety zones are intersecting, hence making the scenario unsuccessful by the definition of the success map, but the boxes representing the vehicles are still not intersecting with each other.

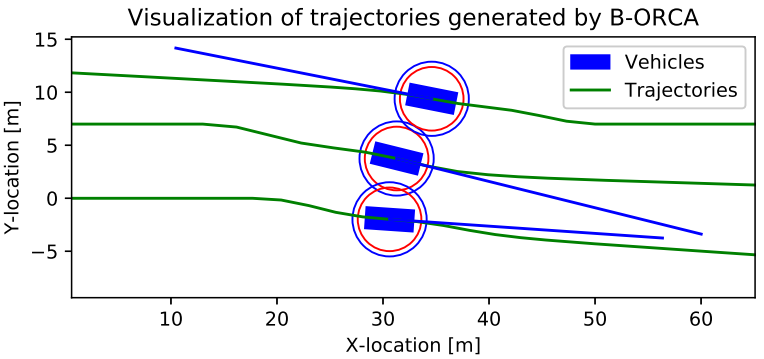

Figure 5.2 Zoomed in view of the trajectories generated by the B-ORCA algorithm in the overtake scenario before they are tracked by the driving robot. The scenario input parameter configuration is $v = 27$m/s and $d = -2$m. Observe that the safety zones are intersecting but the boxes representing the vehicles are not.

Simulation result

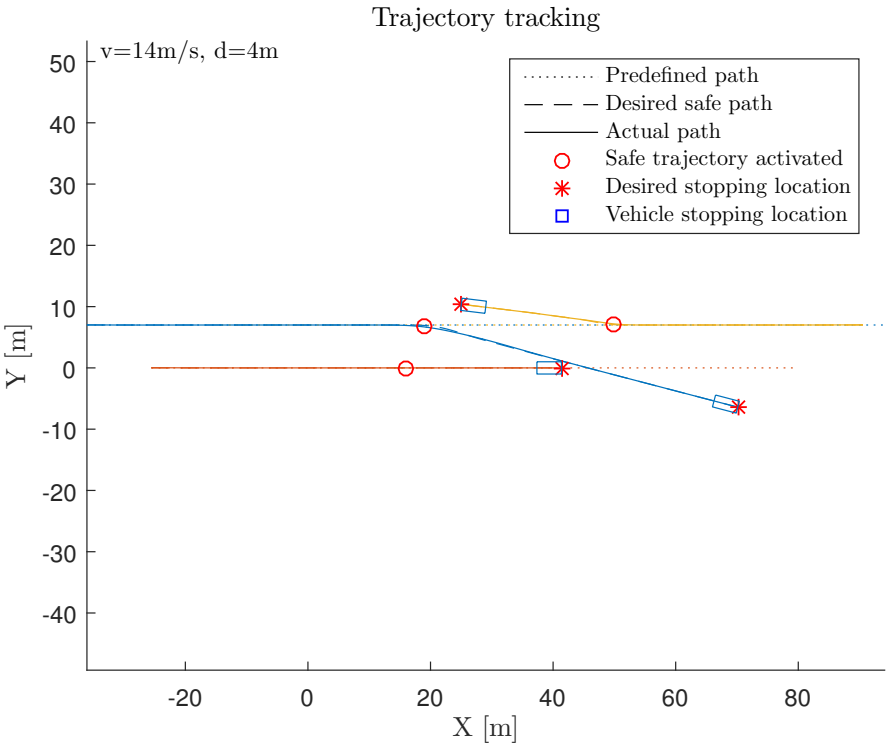

Figure 5.3 Bird's view of the overtake scenario simulation result with initial parameters $v = 14$m/s and $d = 4$m. The color code of the data in this figure is in unison with the colors of the data in the other figures of this test, which makes it easy to follow the data that corresponds to the different vehicles. Blue (Vehicle 1), red (Vehicle 2) and yellow (Vehicle 3).

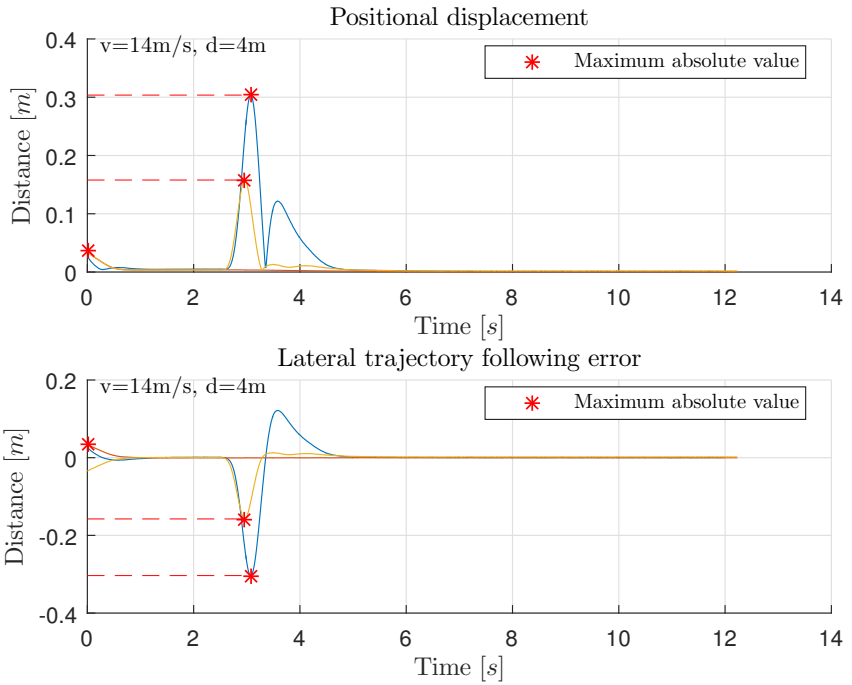

Figure 5.4 Positional displacement and perpendicular path following error of the overtake scenario simulation result with initial parameters $v = 14$m/s and $d = 4$m. The color code of the data in this figure is in unison with the colors of the data in the other figures of this test, which makes it easy to follow the data that corresponds to the different vehicles. Blue (Vehicle 1), red (Vehicle 2) and yellow (Vehicle 3).

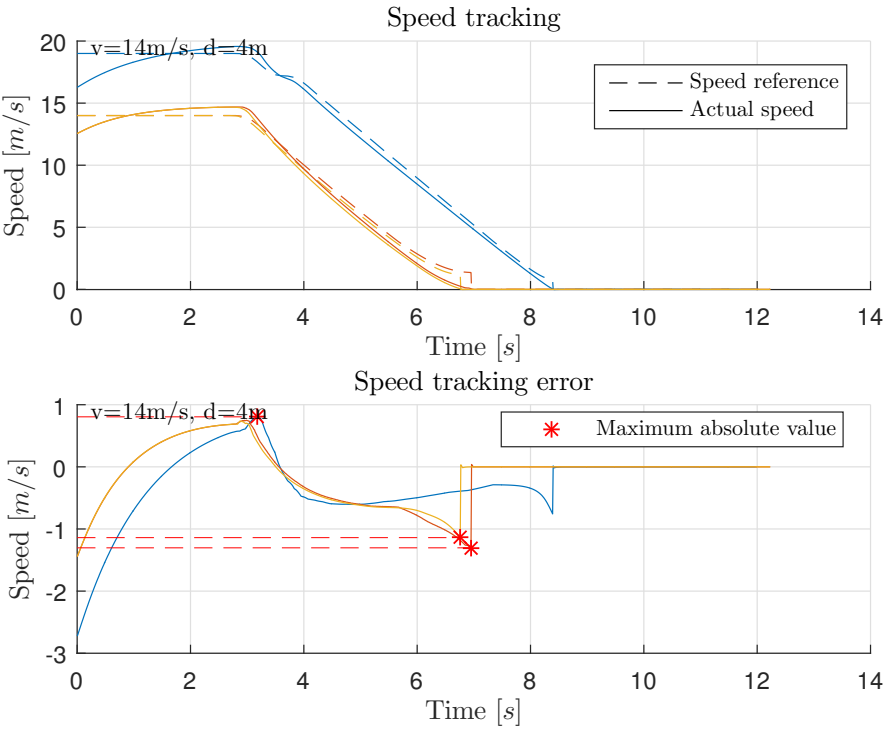

Figure 5.5 Speed following and speed following error of the overtake scenario simulation result with initial parameters $v = 14$m/s and $d = 4$m. The color code of the data in this figure is in unison with the colors of the data in the other figures of this test, which makes it easy to follow the data that corresponds to the different vehicles. Blue (Vehicle 1), red (Vehicle 2) and yellow (Vehicle 3).

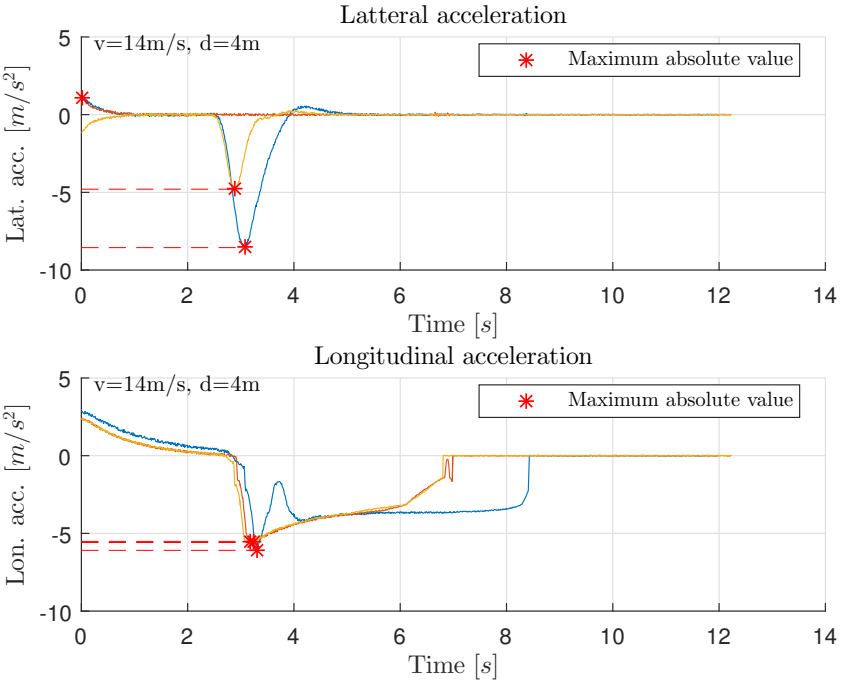

Figure 5.6 Lateral and longitudinal accelerations of the overtake scenario simulation result with initial parameters $v = 14$m/s and $d = 4$m. The color code of the data in this figure is in unison with the colors of the data in the other figures of this test, which makes it easy to follow the data that corresponds to the different vehicles. Blue (Vehicle 1), red (Vehicle 2) and yellow (Vehicle 3).

5.2 Simulation overview

This section gives a total overview of the performance of the B-ORCA algorithm and how easy it is for vehicles to follow the given trajectories. The results of the experiments of all scenarios are combined to give a statistical measure of the average performance with trajectories of different initial speeds and curvatures. This statistical measure serves as an estimation of a bound in which the trajectory following errors and the accelerations of the vehicles can be assumed to be part of, given the current configuration of the B-ORCA algorithm. The values used to give the statistical overview are the maximum of the values for every time series of the different measures presented for example in Section 5.1.

A comparison of the different scenarios will also be presented to show in which scenario the B-ORCA algorithm is more likely to generate dangerous trajectories.

Overall performance

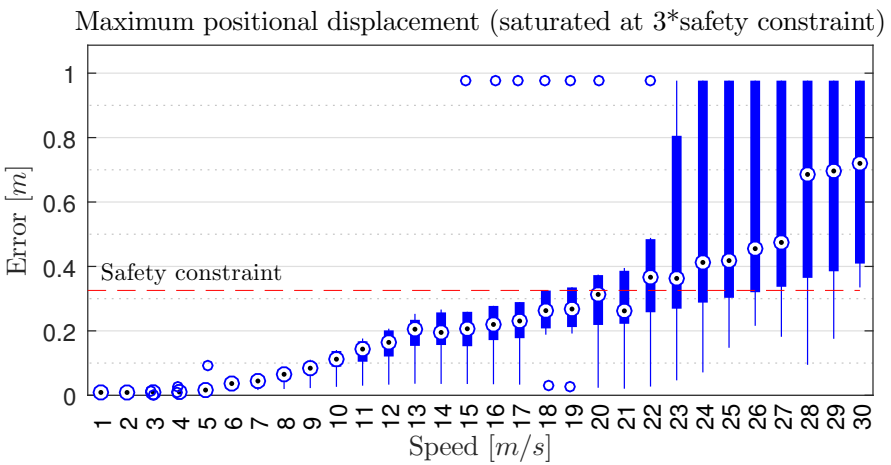

Figure 5.7 Statistical summary of the maximum positional displacement plotted against the initial speed of the vehicles for all scenarios simulated with the Driving robot simulator. The measurements are saturated at 3 times the safety constraint to better capture the performance close to the safety constraint. The statistical average is marked by the target dot on the blue box, which in turn tells the 25 and 75 percentiles. The thin blue lines extending the box are the whiskers that tells how far away from the average the measurements go, and special outliers are marked by the blue circles.

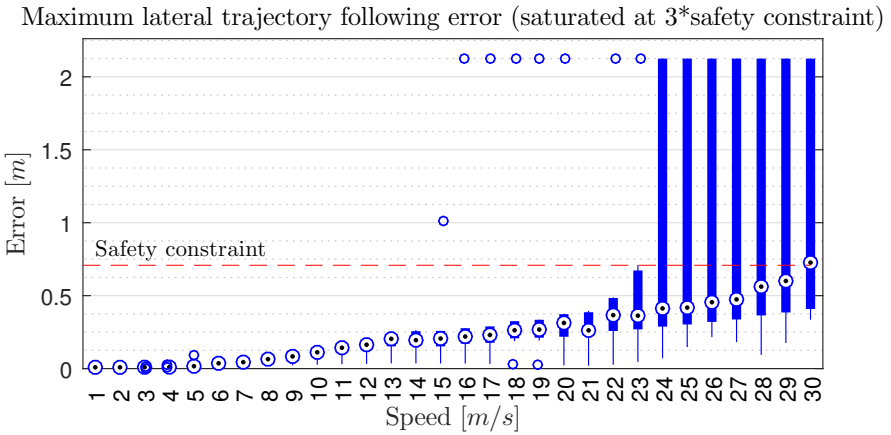

Figure 5.8 Statistical summary of the maximum trajectory following error plotted against the initial speed of the vehicles for all scenarios simulated with the Driving robot simulator. The measurements are saturated at 3 times the safety constraint to better capture the performance close to the safety constraint. The statistical average is marked by the target dot on the blue box, which in turn tells the 25 and 75 percentiles. The thin blue lines extending the box are the whiskers that tells how far away from the average the measurements go, and special outliers are marked by the blue circles.

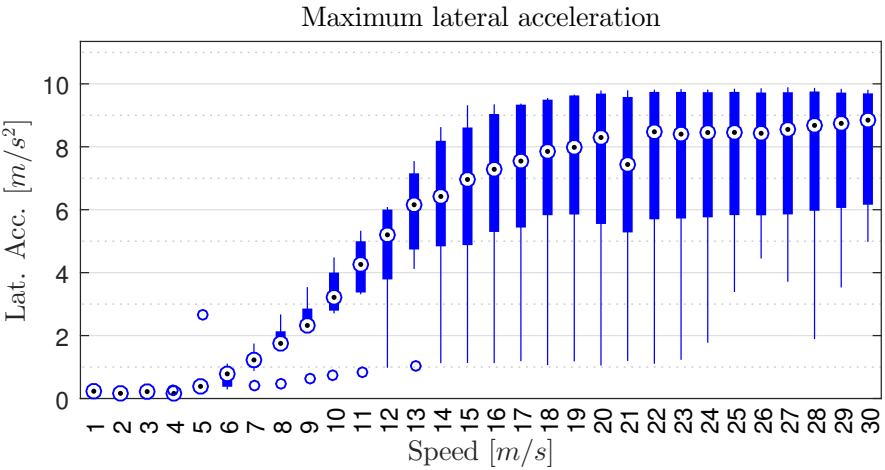

Figure 5.9 Statistical summary of the maximum lateral acceleration plotted against the initial speed of the vehicles for all scenarios simulated with the Driving robot simulator. The statistical average is marked by the target dot on the blue box, which in turn tells the 25 and 75 percentiles. The thin blue lines extending the box are the whiskers that tells how far away from the average the measurements go, and special outliers are marked by the blue circles.

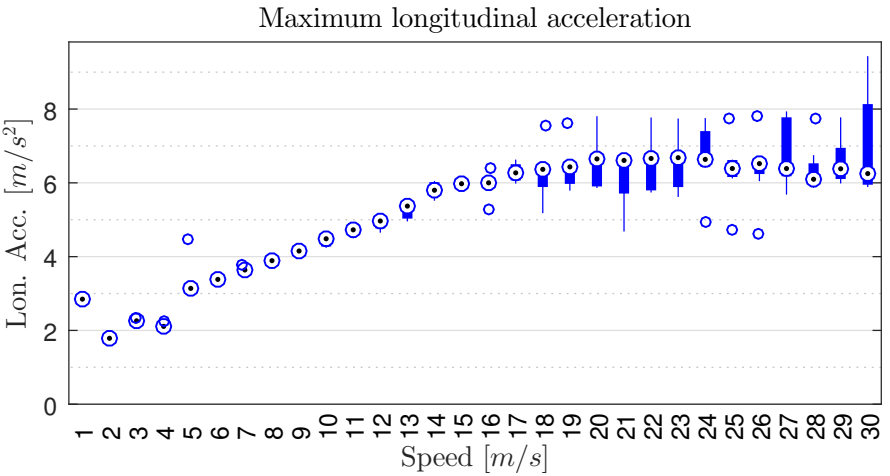

Figure 5.10 Statistical summary of the maximum longitudinal acceleration plotted against the initial speed of the vehicles for all scenarios simulated with the Driving robot simulator. The statistical average is marked by the target dot on the blue box, which in turn tells the 25 and 75 percentiles. The thin blue lines extending the box are the whiskers that tells how far away from the average the measurements go, and special outliers are marked by the blue circles.

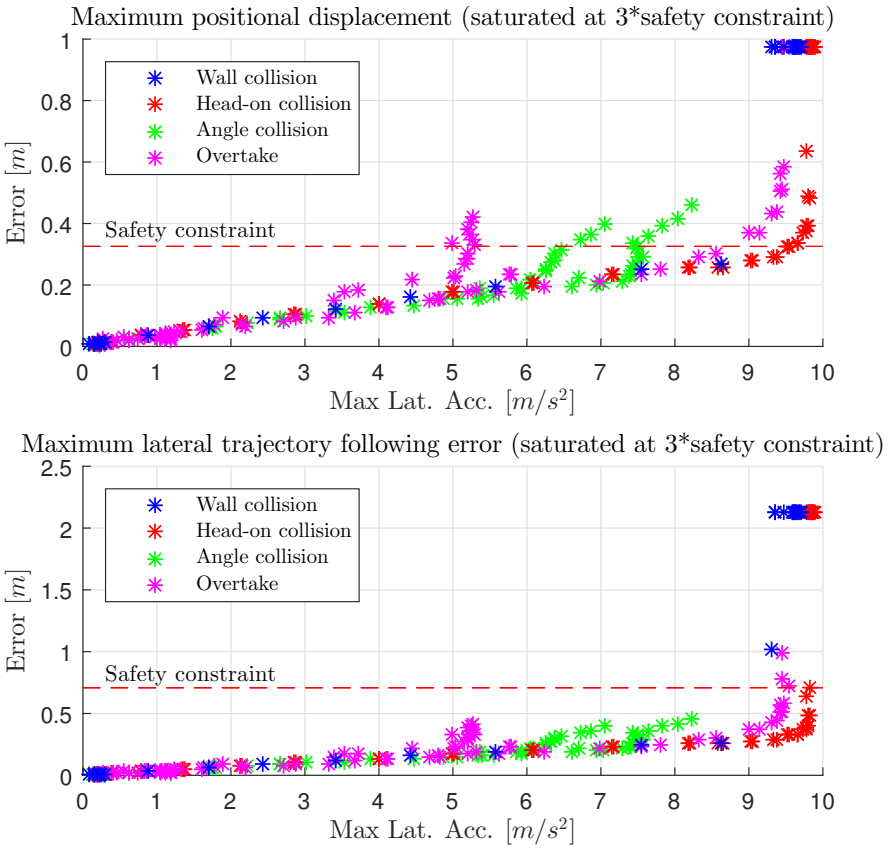

Figure 5.11 Summary of the maximum positional displacement and the trajectory following error plotted against the maximum lateral acceleration of the vehicles for all scenarios simulated with the Driving robot simulator. The measurements are saturated at 3 times the safety constraint to better capture the performance close to the safety constraint.

Scenario comparison

This section contains plots based on the same data as in the previous section, but instead of statistical measures all data points are shown and are separated by colors indicating different scenarios.

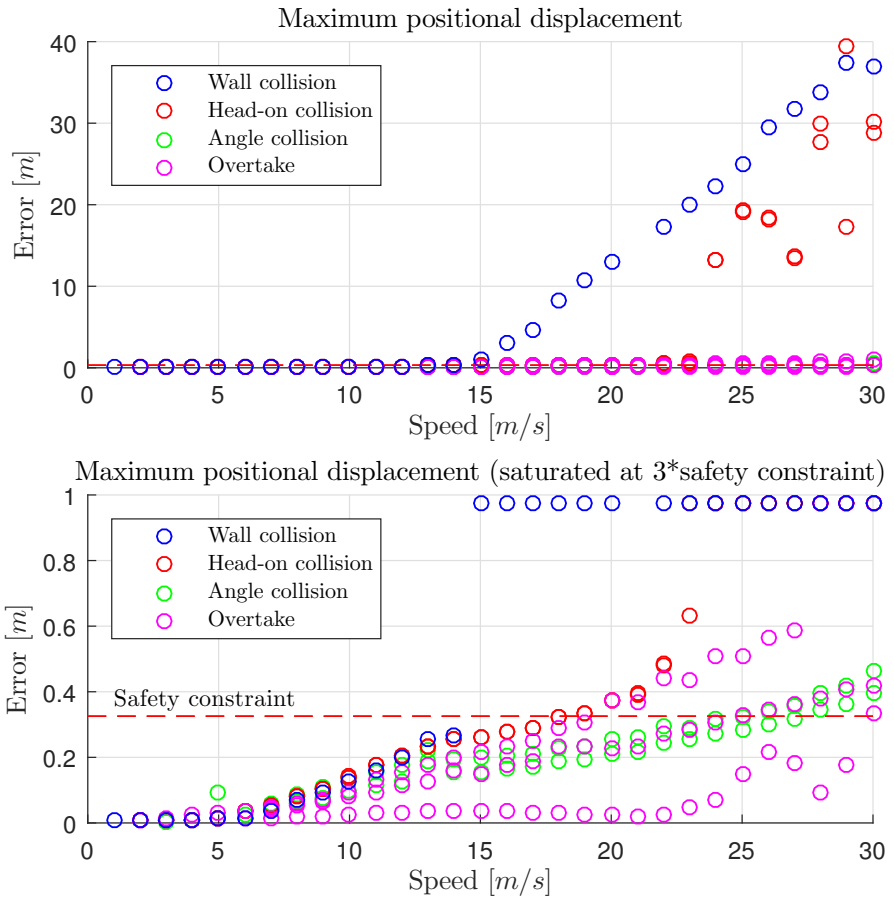

Figure 5.12 Comparison of the maximum positional displacement of the different scenarios when simulated on the Driving robot simulator.

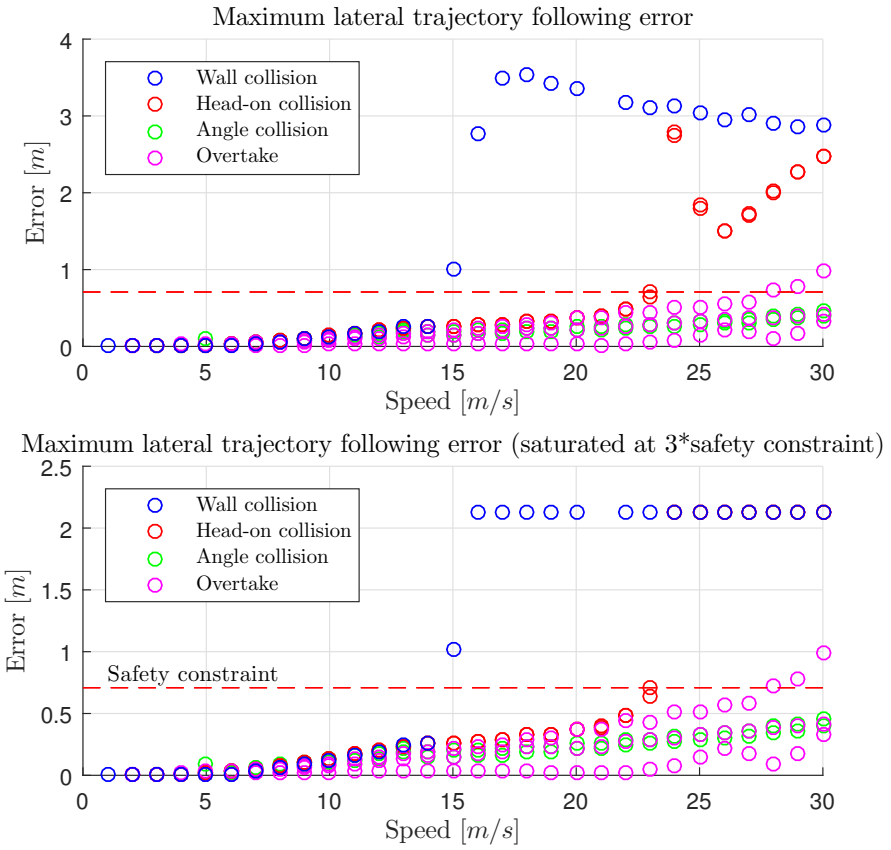

Figure 5.13 Comparison of the maximum lateral path following error of the different scenarios when simulated on the Driving robot simulator.

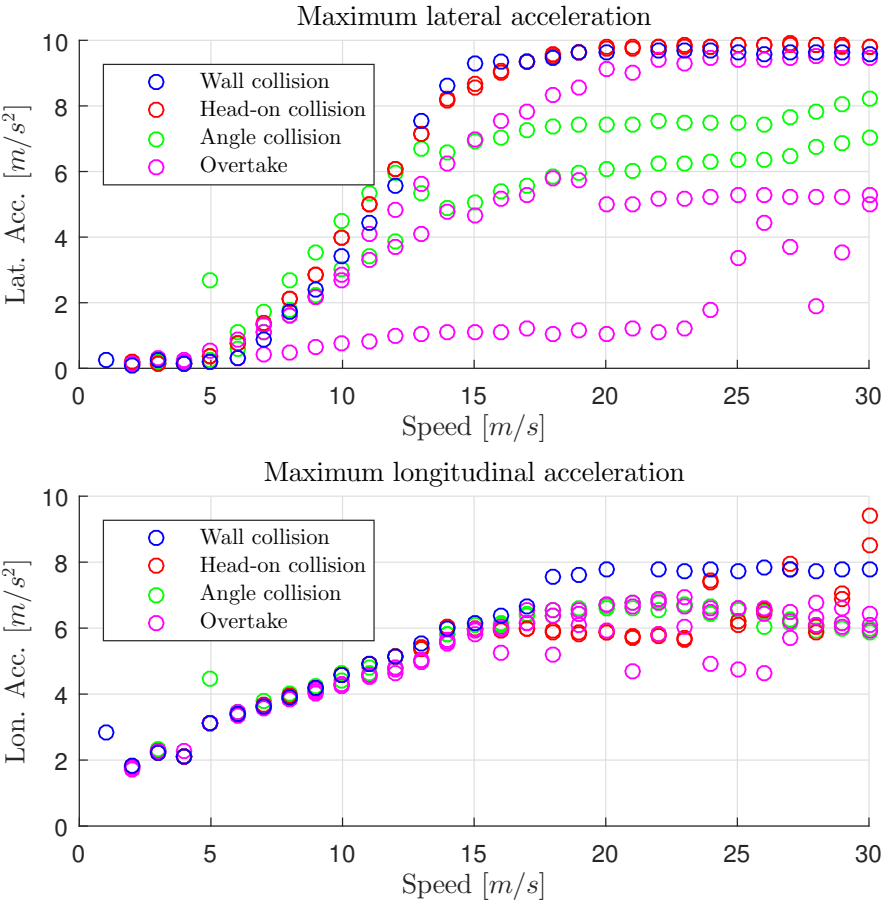

Figure 5.14 Comparison of the maximum lateral and longitudinal acceleration of the different scenarios when simulated on the Driving robot simulator.

5.3 Real vehicle tests

Different configurations of the most critical scenarios, wall collision and head-on collision, were also run on the test track with the real car to compare the edge cases against the simulated result. The result of the tests with the real car is compared to the simulated tests in the figures in this section. The simulated data is the same as in the previous sections of this chapter, and the real data is plotted in the same graph.

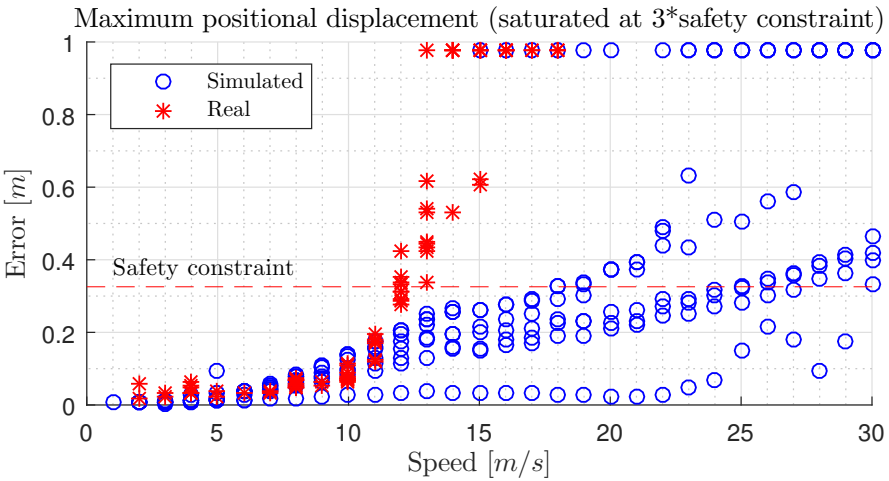

Figure 5.15 Comparison of the maximum positional displacement of the real and simulated tests of the driving robot.

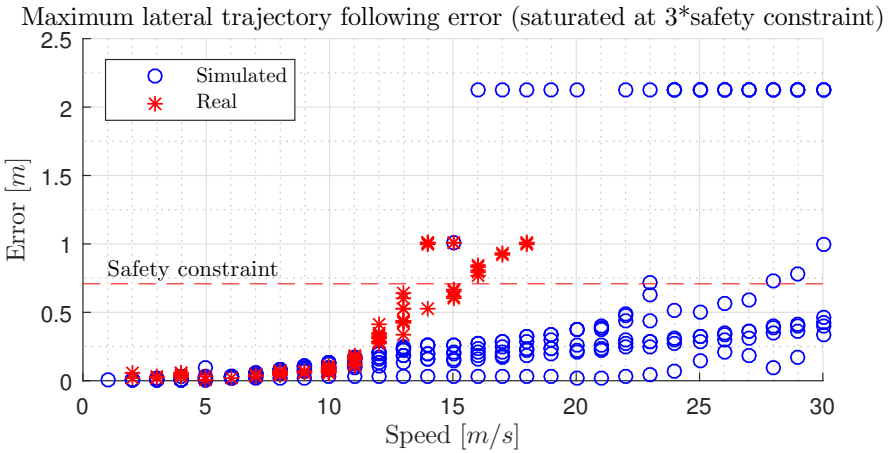

Figure 5.16 Comparison of the maximum lateral path following error of the real and simulated tests of the driving robot.

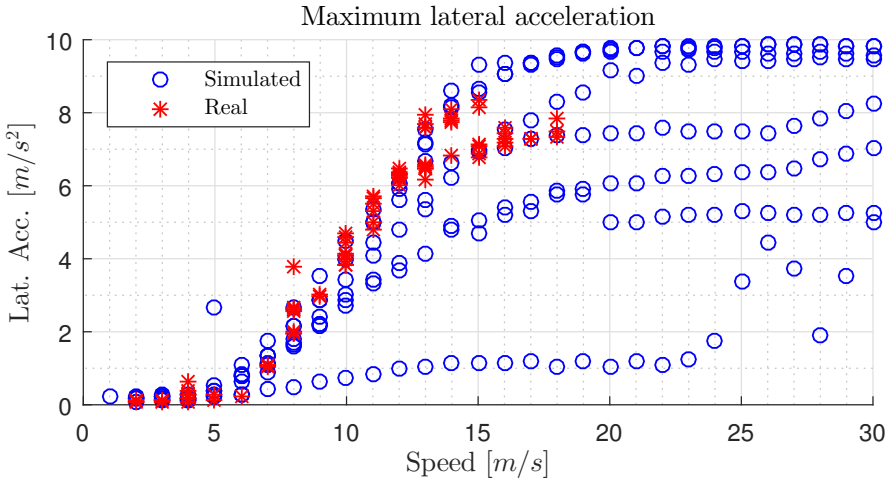

Figure 5.17 Comparison of the maximum lateral acceleration of the real and simulated tests of the driving robot.

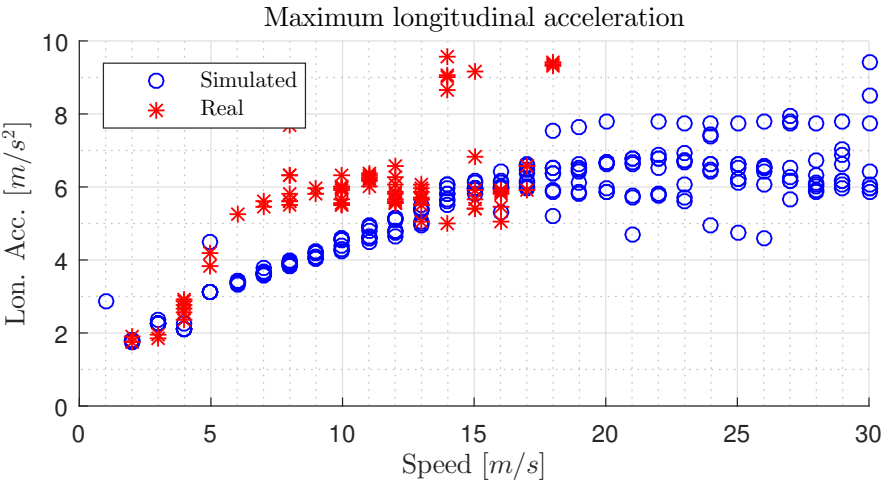

Figure 5.18 Comparison of the maximum longitudinal acceleration of the real and simulated tests of the driving robot.

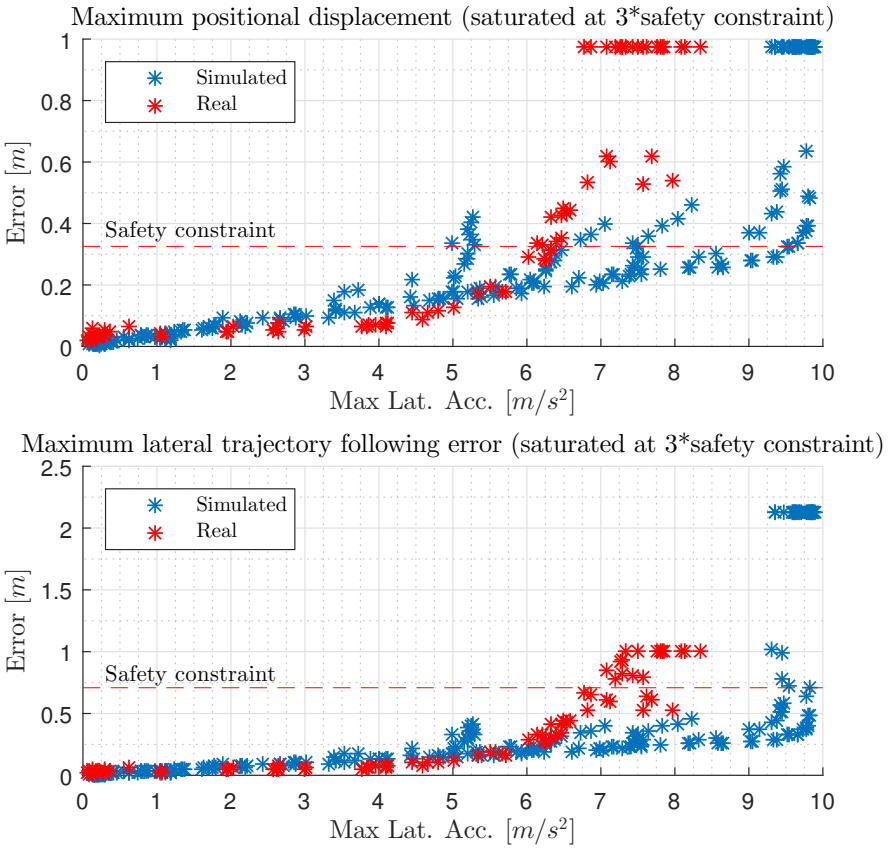

Figure 5.19 Comparison of the summary of the maximum positional displacement and the trajectory following error plotted against the maximum lateral acceleration of the vehicles of all simulations and real tests with the driving robot. The measurements are saturated at 3 times the safety constraint to better capture the performance close to the safety constraint.

6

Discussion

This section covers the discussion of the results presented in Chapter 5.

6.1 Overtake scenario simulations

Success of trajectory generation

The success map of the B-ORCA algorithm's trajectory generation in Figure 5.1 gives a clear overview of which configurations the algorithm succeeds in calculating safe trajectories. For the overtake scenario this figure shows that the algorithm fails in two areas; when the overtaking vehicle is very close to the oncoming vehicle, and when the overtaking vehicle drives with a high speed and is almost side by side with the slower vehicle. This result is reasonable as these should also be the hardest configurations of the scenario to get out of, as it would for any human driver as well. Due to the non-holonomic constraints, there exist configurations from which a collision is physically impossible to avoid. A configuration of a scenario in which it is physically impossible to avoid a collision is for example when two vehicles are on collision course with high speed and are separated with only a short distance when they should activate the maneuver to avoid the collision. As mentioned, this is an impossible situation to avoid a collision in, and the task is therefore to trigger the safety maneuver before getting into that situation. What could be more desirable from the result in Figure 5.1 is a higher success rate when the vehicle 1 is side by side with the slower vehicle 2. The reason for failures at high speeds is due to the overtaking vehicle has nowhere to go, as it is driving too fast to slow down behind the slower car and the oncoming vehicle drives too fast.

Figure 5.2 shows that even if the success map of the scenario says that the B-ORCA algorithm does not produce any safe trajectories for a specific configuration, an absence of safe trajectories is still not guaranteed. That is because the safety zones of the vehicles can be intersecting, but as the safety zones are circular and contain smaller rectangles inside representing the vehicles, the rectangles inside do not have to intersect. Therefore, if any intersection of the safety zones happen, a further analysis can be done to see if the vehicles inside the safety zones will really

collide. The measures of the trajectory generation success given by the success maps are therefore conservative.

Simulation result

In this subsection, the simulation results of the overtake scenario with the configuration $v = 14$ m/s and $d = 4$ m presented in Figures 5.3, 5.4, 5.5 and 5.6 are reviewed. The bird's view of the scenario presented in Figure 5.3 shows that the vehicles follow the desired safe trajectories very well after activation and they also stop at the desired locations. Figure 5.4 shows the time series of the measures of positional displacements and lateral trajectory following. Here it is observed that the errors are larger during the turning sections of the maneuvers, which means that the risk of leaving the safety zones is largest in curvatures of the trajectories. The speed tracking profiles in Figure 5.5 show that the driving robot controller does a good job in tracking the desired velocity, which is very important to stay inside the safety zone defined by the B-ORCA algorithm. The lateral acceleration profiles in Figure 5.6 show how the side forces affecting the vehicles in the curvatures are correlated in time to when the largest errors in the trajectory following presented in Figure 5.4 occur. It is therefore safe to say that the largest errors of the trajectory following occur when the vehicles are turning. The longitudinal acceleration profile in Figure 5.6 shows that the driving robot follows the maximum deceleration of -4m/s^2 specified in the kinematic vehicle model in the B-ORCA algorithm relatively well. There is a small overshoot but the deceleration is then stabilized around -4m/s^2.

6.2 Simulation overview

The figures in Section 5.2 give a good picture of the resulting overall performance of the trajectories generated by the B-ORCA algorithm. Figure 5.7 shows the statistical measure of how the maximum positional displacement increases with speed when following a trajectory. From this figure, it is possible to draw the conclusion that in most cases the driving robot is likely to succeed in following safe trajectories activated with an initial speed of 19m/s (68km/h), with the current configuration of the B-ORCA algorithm. There are outliers violating the constraints already at 15m/s (54km/h), which correspond to the trajectories with high curvature from the wall collision scenario (seen in Figure 5.12). The trajectory for a wall collision scenario can be seen in Figure A.2. When the speed reaches 23m/s it is possible to see that it is very likely that the constraint on positional displacement will be violated, which is due to loss of control as the car loses traction. Staying inside the safety zone is therefore very likely with initial speeds up to 15m/s, after that the vehicles can lose control if the trajectories involve sharp turns but still be able to follow easier curvatures up to 19m/s.

By analyzing the maximum lateral trajectory following error in Figure 5.8, it is possible to observe that the safety constraint is most likely to be violated at 24m/s

(86km/h). It is therefore easier to follow the trajectories when the exact location at given times is not a priority, but rather just the displacement perpendicular to the trajectory. In this figure it is also possible to observe outliers violating the safety constraint already at 15m/s. From Figure 5.13 it is possible to observe that these outliers are the result of the failures when following high curvatures in the wall collision scenario.

Figure 5.9 shows a clear dependency of the statistical measure of the maximum lateral acceleration when plotted against the speed. The maximum lateral acceleration starts by increasing, and then seems to flatten out after 15m/s to not reach a value larger than 10m/s^2. This is interesting when cross referenced to the result in Figure 5.11, as it tells that the main reason for violating the safety constraints is due to high lateral acceleration. Figure 5.9 shows that the the highest lateral accelerations (10m/s^2) start showing in the range 15m/s to 19m/s, which is the range where the safety constraints start getting violated in Figure 5.7. The conclusion from this reasoning is therefore that the main reason for violation of safety constraints is high lateral accelerations which can lead to the vehicle losing traction, which with the current configuration of the B-ORCA algorithm is likely to happen after 15m/s to 19m/s.

In Figure 5.11 there are some measurements of the maximum positional displacement that violate the safety constraints even if they are not close to the maximum lateral acceleration of 10m/s^2. The measurements correspond to the angle collision and overtake scenarios. Cross referencing these measurements to Figure 5.12, there are some measurements at high speeds for the angle collision and overtake scenarios that violate the safety constraints even though they do not seem to have lost control. This indicates that the safety constraints can be violated at very high speeds as well, even though the the lateral acceleration is not that high. The probable cause of these findings is that the driving robot controller starts turning earlier to be able follow a curvature at high speeds, to the cost of more deviation from the desired trajectory.

6.3 Real vehicle tests

In Figure 5.15 it is possible to see that the edge cases of the real tests follows the trend of the simulated cases relatively well. The safety constraint is though violated earlier at around 13m/s instead of 15m/s as is the case for the simulated tests. Similar observations can be seen for the maximum lateral trajectory following error in Figure 5.16, where the safety constraints is violated from around 14m/s.

The probable explanation for this phenomenon can be found in the figures 5.17 and 5.19. As seen for the maximum lateral acceleration plotted against the speed in Figure 5.17, the edge cases of the real test results follows the simulated result accurately, which means that the maximum lateral acceleration of the real tests can be expected to be in the same range as in the simulations at different speeds. As

discussed in Section 6.2, the most probable cause of violating the safety constraints is due to high lateral acceleration. Comparing the real and simulated results of the trajectory following errors in Figure 5.19, it is possible to observe that the violation of the safety constraint for the real tests happens already in the range of $7\text{m}/\text{s}^2$ to $8\text{m}/\text{s}^2$, and the violation of the simulations are in the range $9\text{m}/\text{s}^2$ to $10\text{m}/\text{s}^2$. Therefore, violation of the safety constraints at lower lateral accelerations means that the violation is likely to happen at lower speeds, which is possible to observe in the figures 5.15 and 5.17.

When running the tests on the real vehicle, the ABS and ESC were triggered when the car was doing the maneuvers at higher speeds and accelerations. The activation of these safety systems is a likely source of the difference in behavior of the real and simulated tests, which is seen in the figures mentioned above where the violation of the safety constraints happen at lower maximum lateral accelerations (Figure 5.19).

7

Conclusions and Future work

7.1 Conclusions

In this thesis, an algorithm for collision avoidance of multiple agents in a collaborative way was investigated, implemented and evaluated to be used in a testing framework with driving robots for autonomous driving verification. To achieve the desired objectives, the proposed algorithm was implemented and a set of experiments were set up to evaluate the performance. The calculated safe trajectories were then followed by simulated and real vehicles controlled by driving robots to verify their quality and viability.

Collision avoidance

The proposed B-ORCA algorithm shows promising results in generating safe trajectories for collision avoidance of multiple agents with non-holonomic constraints. The success maps, e.g. Figure 5.1, presented for the benchmarked scenarios show a high success rate for safe trajectory generation. The failures of generating safe trajectories occur when the scenario is critical and because the physical models impose constraints on the vehicles' movements. It is therefore important to activate and follow the safe trajectories before these critical scenarios occur. This can be done by pre-calculating the safe trajectories and having a lookup table similar to the success map in Figure 5.1. Other methods to activate the safe trajectories before a critical scenario occurs include the work done in the master thesis run in parallel to this thesis at Volvo Car Corporation [Abdelwahab, 2019]. That work revolves around calculating the last point to steer or brake in reference to a future collision point of two vehicles.

The proposed B-ORCA is therefore a good choice of algorithm to calculate safe trajectories for safe exit of vehicles. One of the main strengths of the algorithm is the generality of the algorithm as the input states can represent any scenario, from which the algorithm is likely to succeed in producing safe trajectories.

Quality of generated trajectories

The B-ORCA algorithm uses the kinematic bicycle model as a non-holonomic motion model for the controllable dynamic objects in the scenarios to generate viable trajectories for vehicles to follow. This model is though just a simplification of all the kinematic and dynamics of a real vehicle. A deviation from the desired performance for a vehicle to follow the generated safe trajectories well can therefore be expected. A large part of the experiments of this thesis have therefore revolved around identifying when the model can no longer generate viable trajectories. The simulation results presented in Section 5.2 show that the kinematic bicycle model can be used safely in scenarios with speeds until 15m/s (54km/h). Scenarios with speeds above that can still be viable, but speeds of 23m/s (82km/h) and above are very likely to give trajectories that make the vehicle to lose control. The results when tested on real vehicles on the test track are very similar to the simulations, but due to standard safety systems of real cars, large deviations from the desired path were observed already in scenarios with speeds of 13m/s (46km/h).

The constraints chosen to classify safe trajectories are conservative to absolutely guarantee collision free trajectories, but there are still many occasions when the safety constraints are violated but the calculated trajectories still avoid an imminent collision. This can for example happen when the safety zones are intersecting but the vehicles are still not colliding, or when two vehicles violates the safety constraints by cutting corners but by doing so they increases the distance between each other. It can therefore be useful to still use the proposed B-ORCA algorithm to calculate the safe trajectories even if the safety constraints are violated, but perform a separate risk analysis to guarantee their safety.

Reflection

Reflecting back on the research question stated in Section 1.3, this thesis has experimentally verified that the proposed B-ORCA algorithm is able to generate collision-free trajectories for multiple vehicles in a wide array of scenarios where collisions are imminent. The trajectories are calculated with the help of the simple kinematic bicycle model, which was shown to give viable trajectories up to speeds of 13m/s-15m/s (around 50km/h) before the feasibility of the trajectories start to deteriorate.

7.2 Future work

This Section proposes several possible areas of future development related to the work done in this thesis.

Real-Time implementation

To make use of the proposed B-ORCA algorithm the next step is to include it in a system for vehicle-in-the-loop verification described in the Section 1.2 of this thesis. Currently, the algorithm is implemented in Python, but to be able to use it in a verification system it should optimally be ported to C. This can be done in two ways, by either manually rewriting the algorithm in C according to what was implemented in Python, or using the *Cython* library [Behnel et al., 2019] that converts Python source code to C code.

B-ORCA improvements

Increasing the performance of the B-ORCA algorithm can be done by better dimensioning the safety zone to fit the shape of the vehicle. In the current implementation, the safety zone is a circle that is placed around a rectangle which symbolizes the vehicle, see Figure 4.11. The circle around the vehicle will give too much unnecessary space on the side of the vehicle, where it actually would be safe to have another vehicle in. The improvement that could be done to give a safety zone that better fits the shape of the vehicle is to replace the circle with an ellipse. Doing that makes it possible for the vehicles to get closer side by side. The B-ORCA algorithm has to be modified for this in the way that the agents are described by ellipses and the velocity obstacles (see Section 2.1) are created with respect to ellipses instead of circles .

Vehicle model

This thesis concluded that the kinematic bicycle model improves the B-ORCA algorithm's calculation of the guaranteed safe trajectories of speeds up to 13m/s-15m/s (approx. 50km/h). This is still relatively fast, and many tests can be run with this configuration. If the tests will be done in higher speeds than that, it is recommended to implement a more sophisticated vehicle model that can set limits on the lateral accelerations which was the main reason for violating the safety constraints. An example of a model that could be used for this is the *Dynamic bicycle model* [Kong et al., 2015] that also takes the forces acting on the tires into consideration.

Bibliography

Abdelwahab, S. (2019). *Last Point to React for Collision Avoidance in a Robotized Framework for Autonomous Driving Verification*. MA thesis. Chalmers University of Technology, Dept. of Electrical Engineering, Gothenburg, Sweden.

Alonso-Mora, J., A. Breitenmoser, P. Beardsley, and R. Siegwart (2012). "Reciprocal collision avoidance for multiple car-like robots". *2012 IEEE International Conference on Robotics and Automation*, pp. 1–16.

Association for safe international road travel (2018). *Road safety facts*. URL: https://www.asirt.org/safe-travel/road-safety-facts/ (visited on 2019-05-29).

Åström, K.-J., K.-E. Årzén, and B. Wittenmark (2002). "Computer control: an overview". *(IFAC PROFESSIONAL BRIEF). International Federation of Automatic Control.*

Behnel, S., R. Bradshaw, L. Dalcín, M. Florisson, V. Makarov, and D. S. Seljebotn (2019). *Cyton: c-extension for python*. URL: https://cython.org/ (visited on 2019-05-29).

Bellis, M. (2018). *The history of airbags*. ThoughtCo. URL: https://www.thoughtco.com/history-of-airbags-1991232 (visited on 2019-05-29).

Bertsekas, D. (2009). *Convex optimization theory*. 3, pp. 487–487. ISBN: 9780521833783. DOI: 10.1109/TAC.2006.884922. arXiv: 1111.6189v1.

Burton, D., S. Newstead, and D. Logan (2004). "Evaluation of anti-lock braking systems effectiveness". URL: https://www.researchgate.net/publication/224709393_Evaluation_of_Anti-lock_Braking_Systems_Effectiveness (visited on 2019-05-29).

Campbell, S. F. (2007). *Steering Control of an Autonomous Ground Vehicle with Application to the DARPA Urban Challenge*. MA thesis. Massachusetts Institute of Technology, Dept. of Mechanical Engineering, Boston, Massachusetts, pp. 487–492.

CBS Insight (2018). *46 corporations working on autonomous vehicles*. CBS Insight. URL: https://www.cbinsights.com/research/autonomous-driverless-vehicles-corporations-list/ (visited on 2019-05-29).

Dijkstra, E. W. (1959). "A Note on Two Problems in Connexion with Graphs". *Springer-Verlag*, pp. 269–271. DOI: 10.1007/BF01386390.

Euro NCAP (2019). *Electronic stability control*. Euro NCAP. URL: https://www.euroncap.com/en/vehicle-safety/the-ratings-explained/safety-assist/esc/ (visited on 2019-05-29).

Fiorini, P. and Z. Shiller (1998). "Motion planning in dynamic environments using the relative velocity paradigm". *The International Journal of Robotics Research, 17(7)*, pp. 760–772. DOI: 10.1177/027836499801700706.

Ghazaei Ardakani, M., B. Olofsson, A. Robertsson, and R. Johansson (2015). "Real-Time Trajectory Generation using Model Predictive Control". *2015 IEEE Conference on Automation Science and Engineering (CASE) 2015*. DOI: 10.1109/CoASE.2015.7294220.

Goldfarb, D. and A. Idnani (1982). *Dual and primal-dual methods for solving strictly convex quadratic programs*. Vol. 909. In: Hennart J.P. (eds) Numerical Analysis. Lecture Notes in Mathematics. Springer, Berlin, Heidelberg.

González, D., J. Pérez, V. Milanés, and F. Nashashibi (2016). "A Review of Motion Planning Techniques for Automated Vehicles". *IEEE Transactions on Intelligent Transportation Systems* **17**:4, pp. 1135–1145. DOI: 10.1109/TITS.2015.2498841.

Hart, P. E., N. J. Nilsson, and R. Bertram (1968). "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". *IEEE Transactions on Systems Science and Cybernetics* **4**:2, pp. 100–107. DOI: 10.1109/TSSC.1968.300136.

Kavralu, L. E., P. Svestka, J.-c. Latombe, and M. H. Overmars (1996). "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces". *IEEE Transactions on Robotics and Automation* **12**:4. DOI: 10.1109/70.508439.

Khatib, O. (1985). "Real-Time obstacle avoidance for manipulators and mobile robots". *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, pp. 500–505. DOI: 10.1109/ROBOT.1985.1087247.

Kong, J., M. Pfeiffer, G. Schildbach, and F. Borrelli (2015). "Kinematic and dynamic vehicle models for autonomous driving control design". *2015 IEEE Intelligent Vehicles Symposium (IV)*, pp. 2–7.

Kuwata, Y., J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How (2009). "Real-time Motion Planning with Applications to Autonomous Urban Driving". *IEEE Transactions on Control Systems Technology* **17**:5, pp. 1105–1118. DOI: 10.1109/TCST.2008.2012116.

LaValle, S. M. (1998). *Rapidly-Exploring Random Trees: A New Tool for PathPlanning*. TR 98-11, Computer Science Dept., Iowa State University.

Likhachev, M., D. Ferguson, G. Gordon, A. Stentz, and S. Thrun (2001). "Anytime Dynamic A*: An Anytime , Replanning Algorithm". *American Association for Artificial Intelligence*.

McCormick, L. W. (2019). *A short history of the airbag*. Consumer Affairs. URL: https://www.consumeraffairs.com/news04/2006/airbags/airbags_invented.html (visited on 2019-05-29).

New York Times (2019). *Prosecutors don't plan to charge uber in self-driving car's fatal accident*. New York Times. URL: https://www.nytimes.com/2019/03/05/technology/uber-self-driving-car-arizona.html (visited on 2019-05-29).

Petereit, J., T. Emter, C. W. Frey, T. Kopfstedt, and A. Beutel (2012). "Application of Hybrid A * to an Autonomous Mobile Robot for Path Planning in Unstructured Outdoor Environments". *ROBOTIK 2012; 7th German Conference on Robotics*, pp. 227–232.

SAE International (2018). "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles (j3016_201806)". *Society of Automotive Engineers*.

Stentz, A. (1994). "Optimal and Efficient Path Planning for Partially-Known Environments". *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pp. 3310–3317. DOI: 10.1109/ROBOT.1994.351061.

Tesla (2016). *All tesla cars being produced now have full self-driving hardware*. Tesla. URL: https://www.tesla.com/sv_SE/blog/all-tesla-cars-being-produced-now-have-full-self-driving-hardware (visited on 2019-05-29).

The Guardian (2018). *Tesla car that crashed and killed driver was running on autopilot, firm says*. The Guardian. URL: https://www.cbinsights.com/research/autonomous-driverless-vehicles-corporations-list/ (visited on 2019-05-29).

Trafikverket (2018a). *Det här är nollvisionen*. Trafikverket. URL: https://www.trafikverket.se/resa-och-trafik/Trafiksakerhet/det-har-ar-nollvisionen/ (visited on 2019-05-29).

Trafikverket (2018b). *Rattfylleri*. Trafikverket. URL: https://www.trafikverket.se/resa-och-trafik/Trafiksakerhet/Din-sakerhet-pa-vagen/Rattfylleri/ (visited on 2019-05-29).

Van Den Berg, J., S. J. Guy, M. Lin, and D. Manocha (2011). "Reciprocal n-body collision avoidance". *Springer Tracts in Advanced Robotics* **70**, pp. 3–19. ISSN: 16107438. DOI: 10.1007/978-3-642-19457-3_1.

Van den Berg, J., M. Lin, and D. Manocha (n.d.). "Reciprocal Velocity Obstacles for Real-Time Multi-Agent Navigation". *2008 IEEE International Conference on Robotics and Automation*. DOI: 10.1109/ROBOT.2008.4543489.

Volvo Car Corporation (2009). *3-point safety belt from volvo - the most effective lifesaver in traffic for fifty years*. URL: https://www.media.volvocars.com/global/en-gb/media/pressreleases/18405 (visited on 2019-05-29).

Zhou, L. and W. Li (2014). "Adaptive Artificial Potential Field Approach for Obstacle Avoidance Path Planning". *2014 Seventh International Symposium on Computational Intelligence and Design* **2**:1, pp. 429–432. DOI: 10.1109/ISCID.2014.144.

A

Results cont.

A.1 Wall collision scenario

Success of path generation

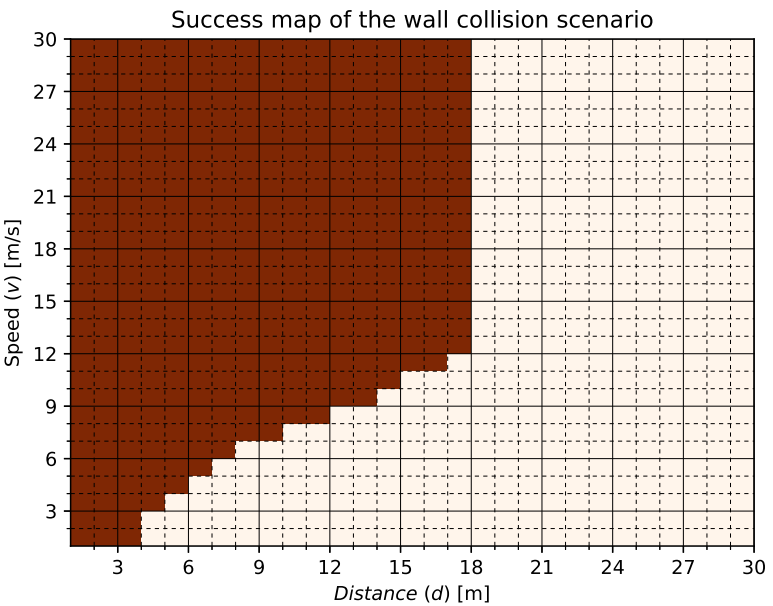

Figure A.1 Success map of the path generation of the wall collision scenario. Red color symbolizes an unsuccessful trajectory generation of the B-ORCA algorithm, and white color symbolizes a successful trajectory generation. The values on the axes are the input parameters of the scenario, defined in Table 4.2

Simulation result

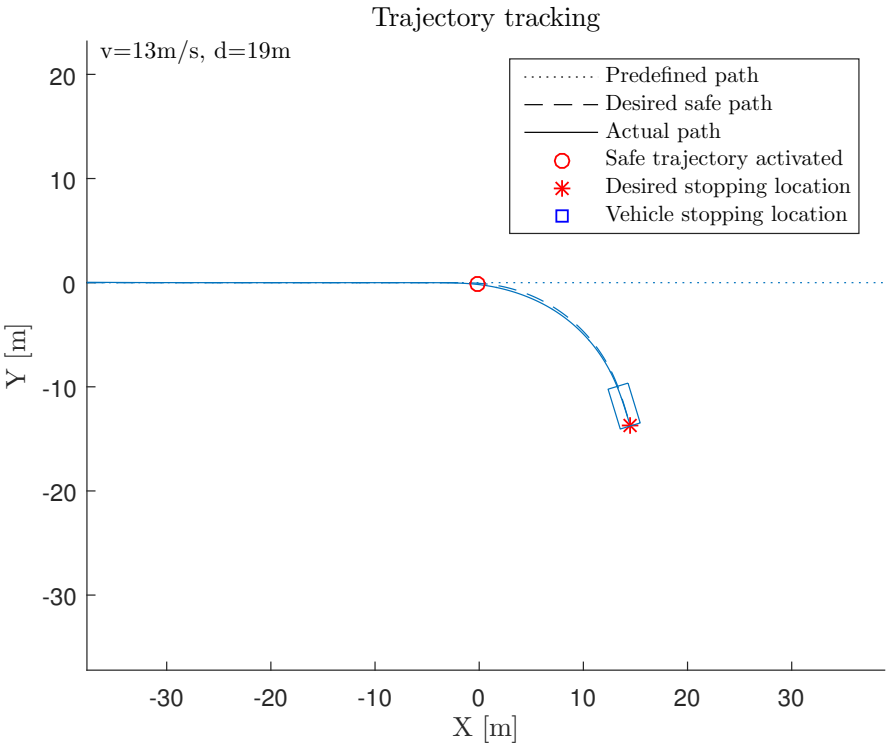

Figure A.2 Bird's view of the wall collision scenario with initial parameters $v =$ 13m/s and $d = 24$m. Simulated on the driving robot simulator.

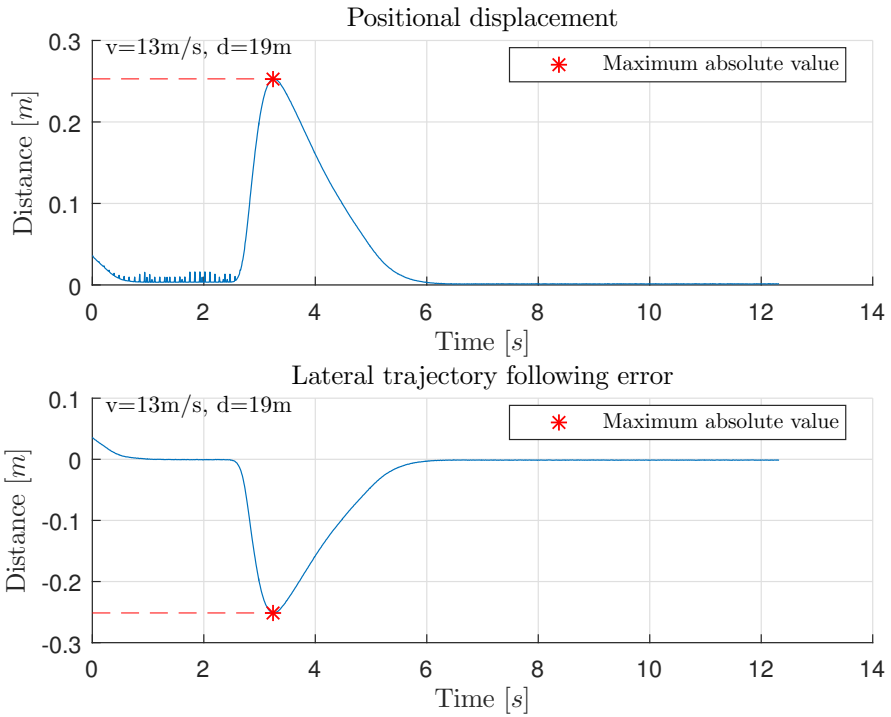

Figure A.3 Positional displacement and perpendicular path following error of the wall collision scenario with initial parameters $v = 13$m/s and $d = 24$m. Simulated on the driving robot simulator.

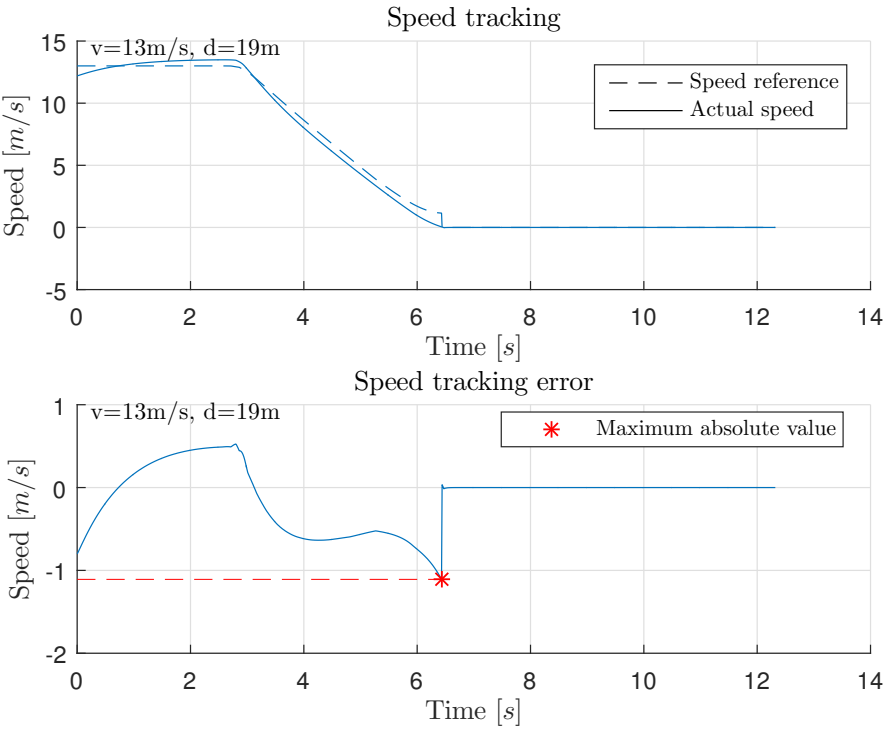

Figure A.4 Speed following and speed following error of the wall collision scenario with initial parameters $v = 13$m/s and $d = 24$m. Simulated on the driving robot simulator.

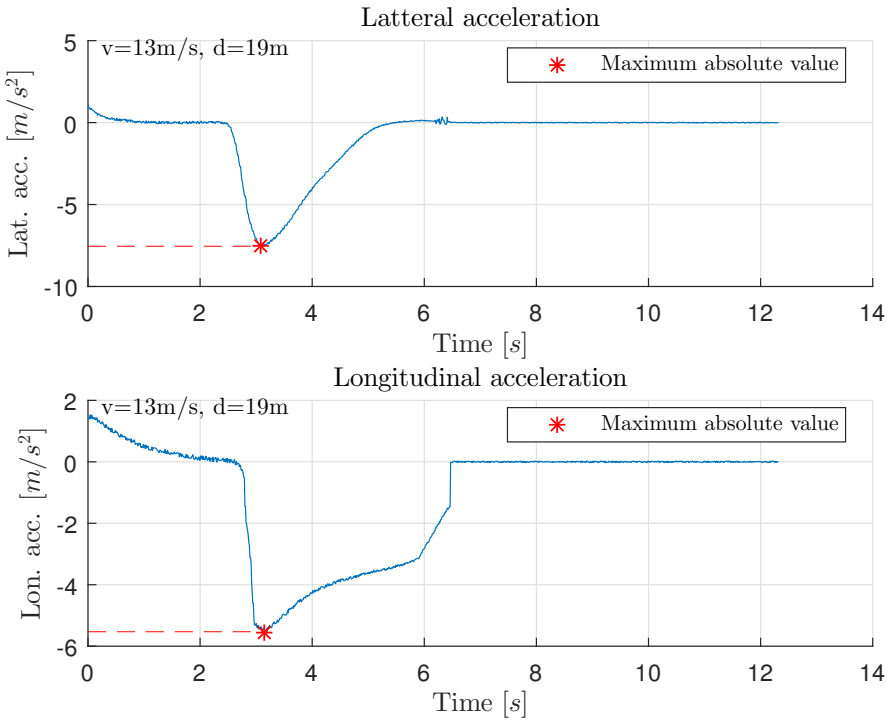

Figure A.5 Lateral and longitudinal accelerations of the wall collision scenario with initial parameters $v = 13$m/s and $d = 24$m. Simulated on the driving robot simulator.

Real vehicle test result

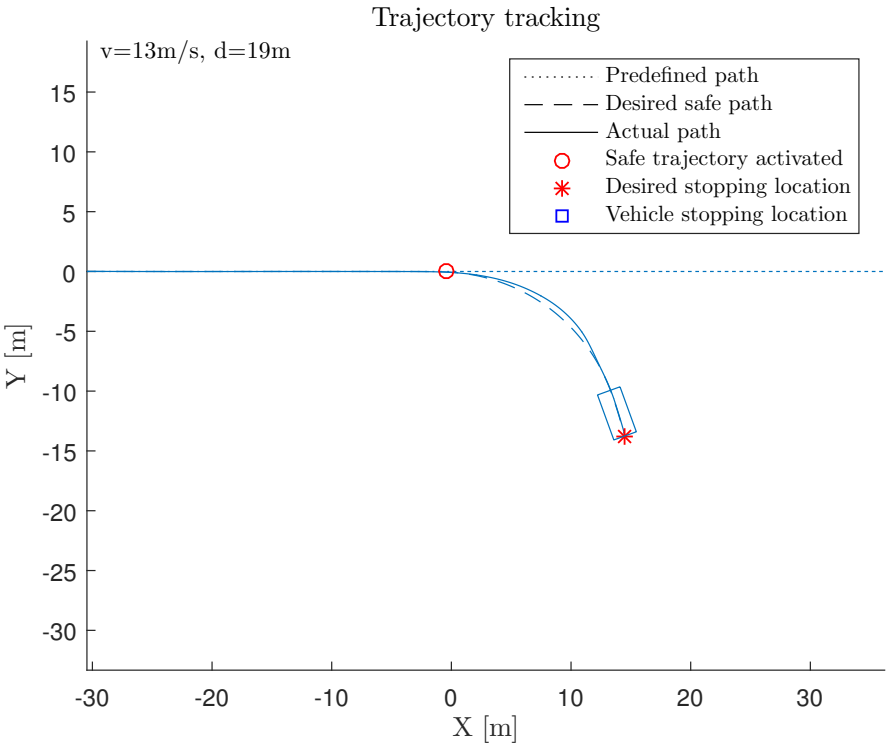

Figure A.6 Bird's view of the wall collision scenario with initial parameters $v =$ 13m/s and $d =$ 19m. Test run on the real driving robot controlling a car.

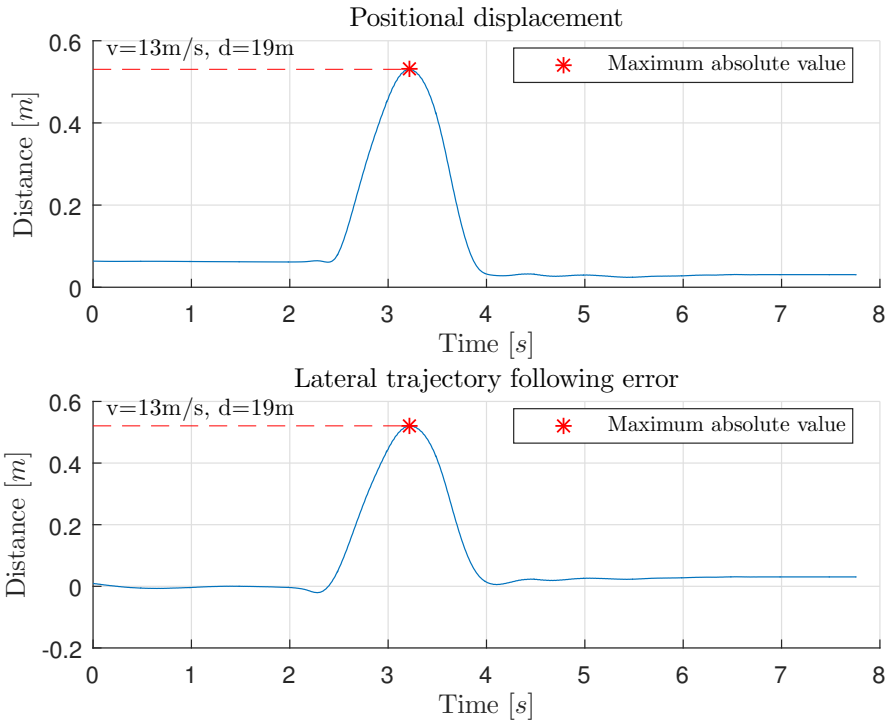

Figure A.7 Positional displacement and perpendicular path following error of the wall collision scenario with initial parameters $v = 13$m/s and $d = 19$m. Test run on the real driving robot controlling a car.

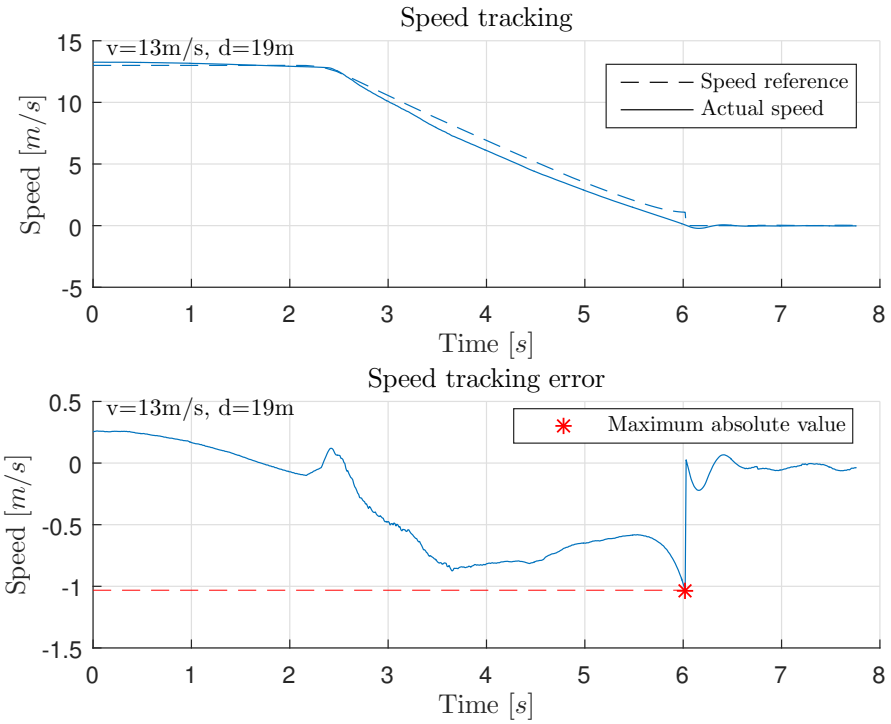

Figure A.8 Speed following and speed following error of the wall collision scenario with initial parameters $v = 13$m/s and $d = 19$m. Test run on the real driving robot controlling a car.

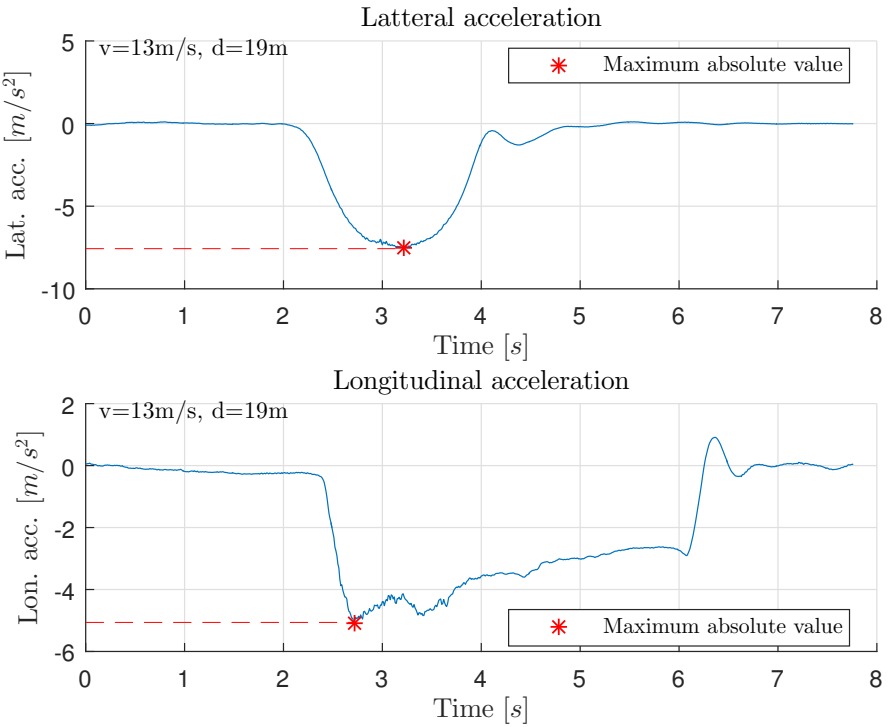

Figure A.9 Lateral and longitudinal accelerations of the wall collision scenario with initial parameters $v = 13$m/s and $d = 19$m. Test run on the real driving robot controlling a car.

A.2 Head-on collision scenario

Success of path generation

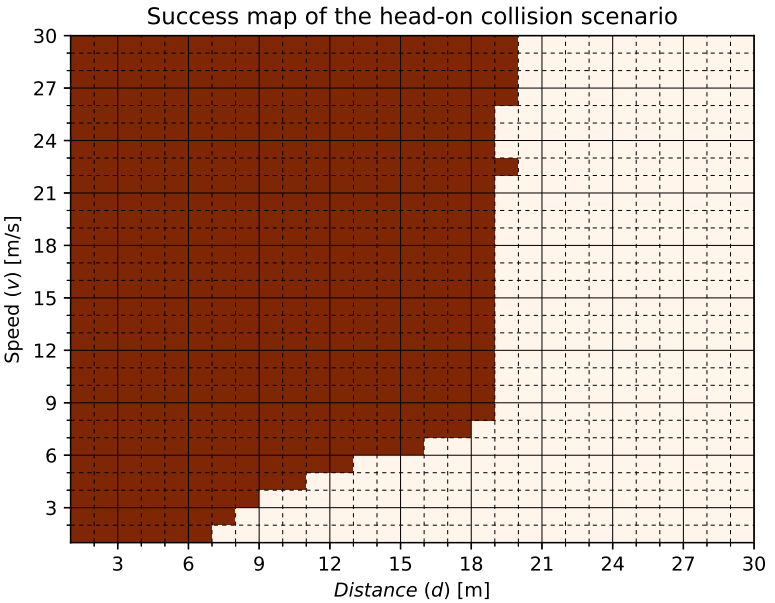

Figure A.10 Success map of the path generation of the head-on collision scenario. Red color symbolizes an unsuccessful trajectory generation of the B-ORCA algorithm, and white color symbolizes a successful trajectory generation. The values on the axes are the input parameters of the scenario, defined in Table 4.3.

Simulation result

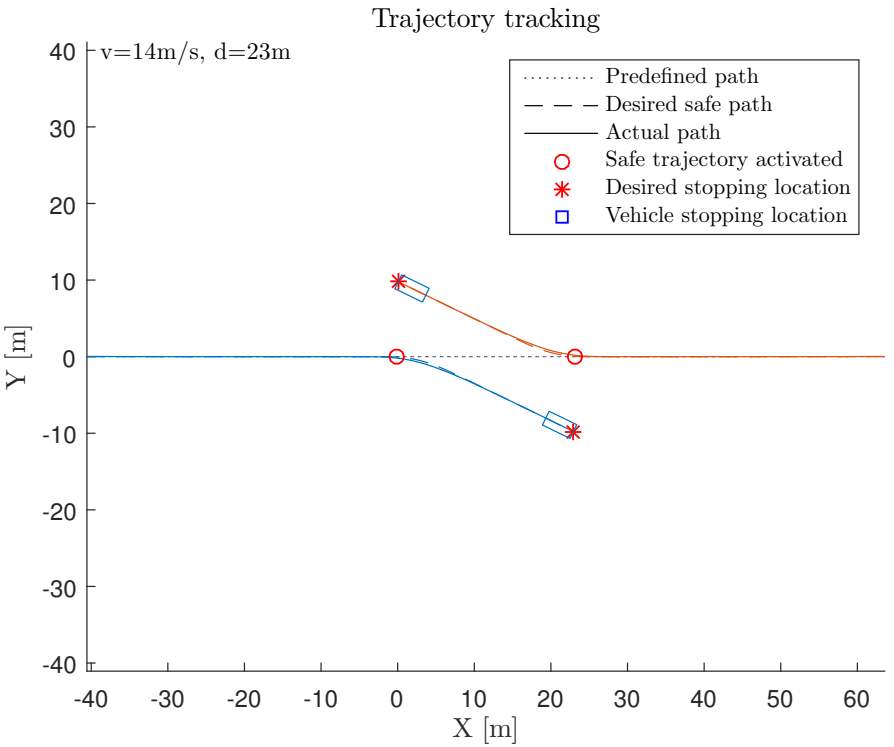

Figure A.11 Bird's view of the head-on collision scenario with initial parameters $v = 14$m/s and $d = 23$m. Simulated on the driving robot simulator. The color code of the data in this figure is in unison with the colors of the data in the other figures of this test, which makes it easy to follow the data that corresponds to the different vehicles. Blue (Vehicle 1), red (Vehicle 2).

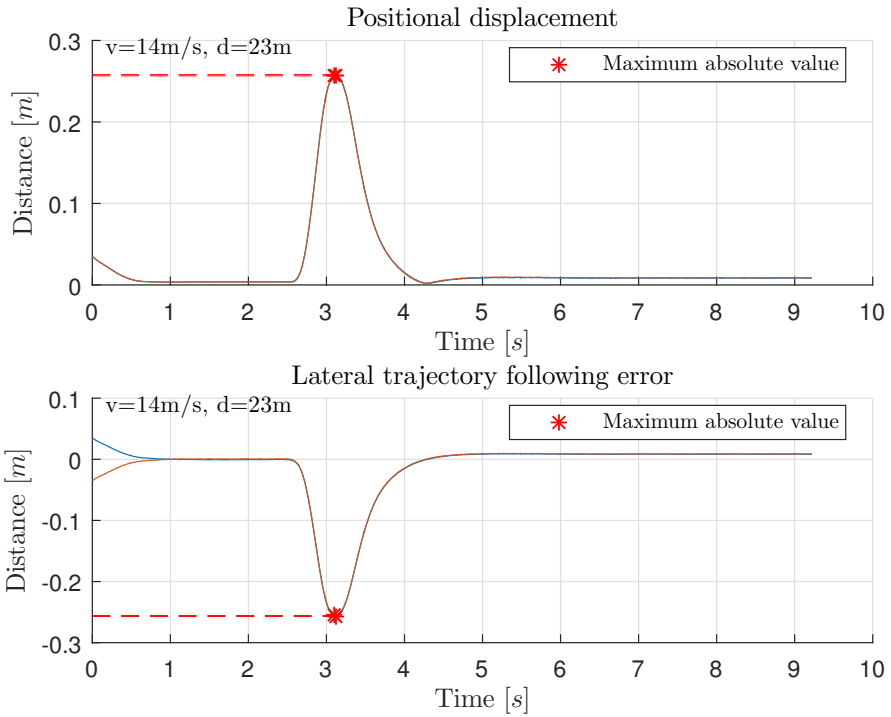

Figure A.12 Positional displacement and perpendicular path following error of the head-on collision scenario with initial parameters $v = 14$m/s and $d = 23$m. Simulated on the driving robot simulator. The color code of the data in this figure is in unison with the colors of the data in the other figures of this test, which makes it easy to follow the data that corresponds to the different vehicles. Blue (Vehicle 1), red (Vehicle 2). Note: The two time-series are plotted on top on each other as the scenario is symmetrical.

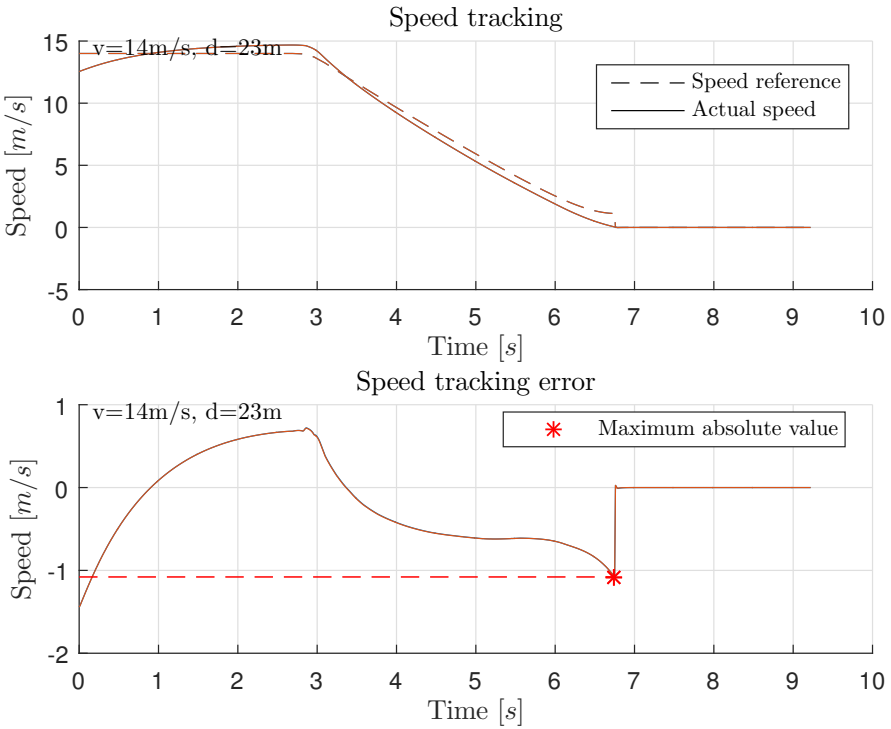

Figure A.13 Speed following and speed following error of the head-on collision scenario with initial parameters $v = 14$m/s and $d = 23$m. Simulated on the driving robot simulator. The color code of the data in this figure is in unison with the colors of the data in the other figures of this test, which makes it easy to follow the data that corresponds to the different vehicles. Blue (Vehicle 1), red (Vehicle 2). Note: The two time-series are plotted on top on each other as the scenario is symmetrical.

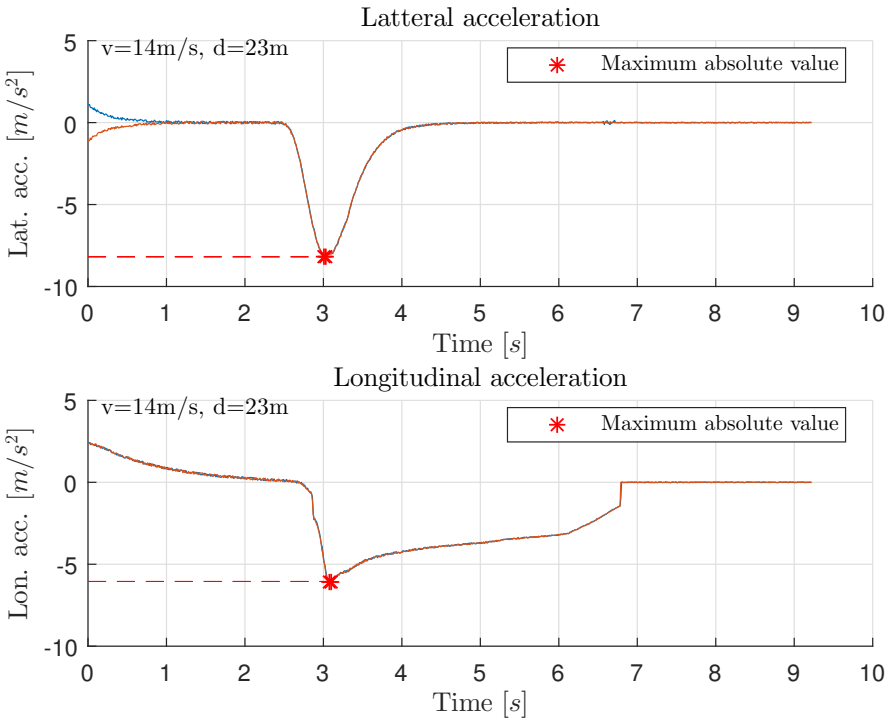

Figure A.14 Lateral and longitudinal accelerations of the head-on collision scenario with initial parameters $v = 14$m/s and $d = 23$m. Simulated on the driving robot simulator. The color code of the data in this figure is in unison with the colors of the data in the other figures of this test, which makes it easy to follow the data that corresponds to the different vehicles. Blue (Vehicle 1), red (Vehicle 2). Note: The two time-series are plotted on top on each other as the scenario is symmetrical.

A.3 Angle collision scenario

Success of path generation

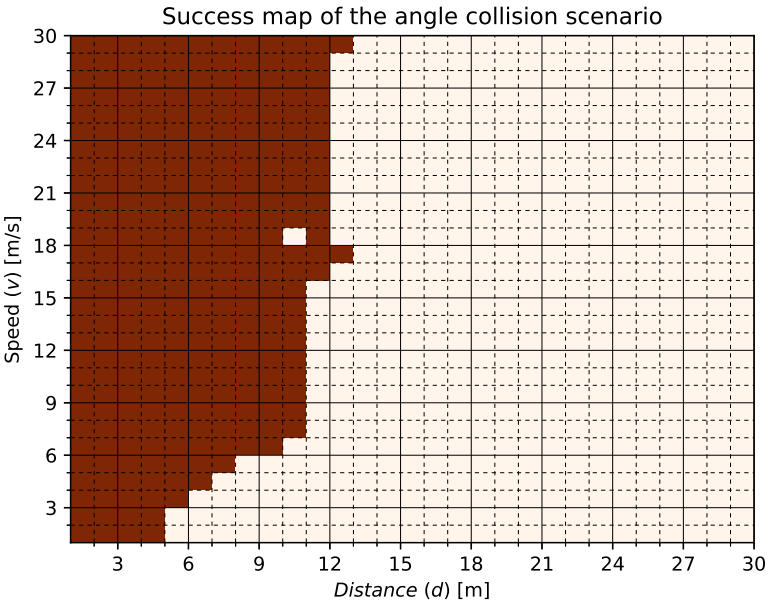

Figure A.15 Success map of the path generation of the angle collision scenario. Red color symbolizes an unsuccessful trajectory generation of the B-ORCA algorithm, and white color symbolizes a successful trajectory generation. The values on the axes are the input parameters of the scenario, defined in Table 4.4.

Simulation result

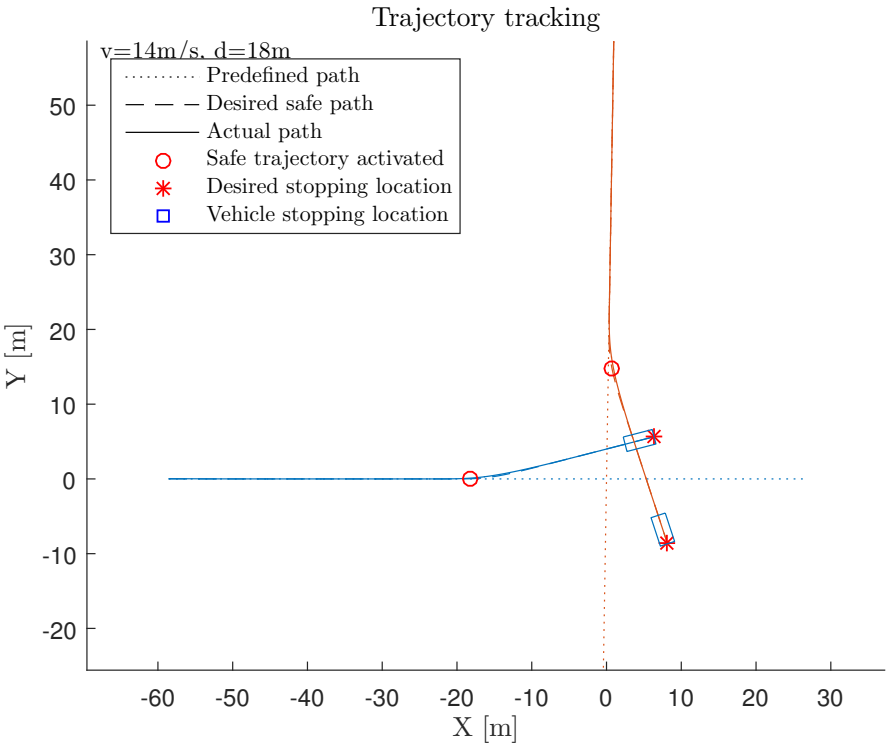

Figure A.16 Bird's view of the angle collision scenario with initial parameters $v = 14$m/s and $d = 18$m. Simulated on the driving robot simulator. The color code of the data in this figure is in unison with the colors of the data in the other figures of this test, which makes it easy to follow the data that corresponds to the different vehicles. Blue (Vehicle 1), red (Vehicle 2).

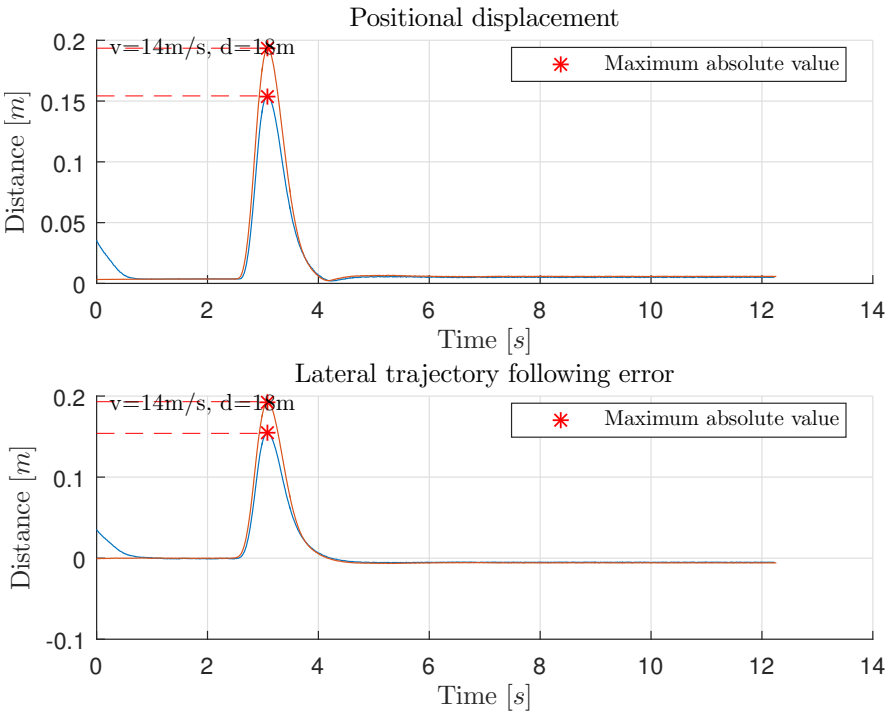

Figure A.17 Positional displacement and perpendicular path following error of the angle collision scenario with initial parameters $v = 14$m/s and $d = 18$m. Simulated on the driving robot simulator. The color code of the data in this figure is in unison with the colors of the data in the other figures of this test, which makes it easy to follow the data that corresponds to the different vehicles. Blue (Vehicle 1), red (Vehicle 2).

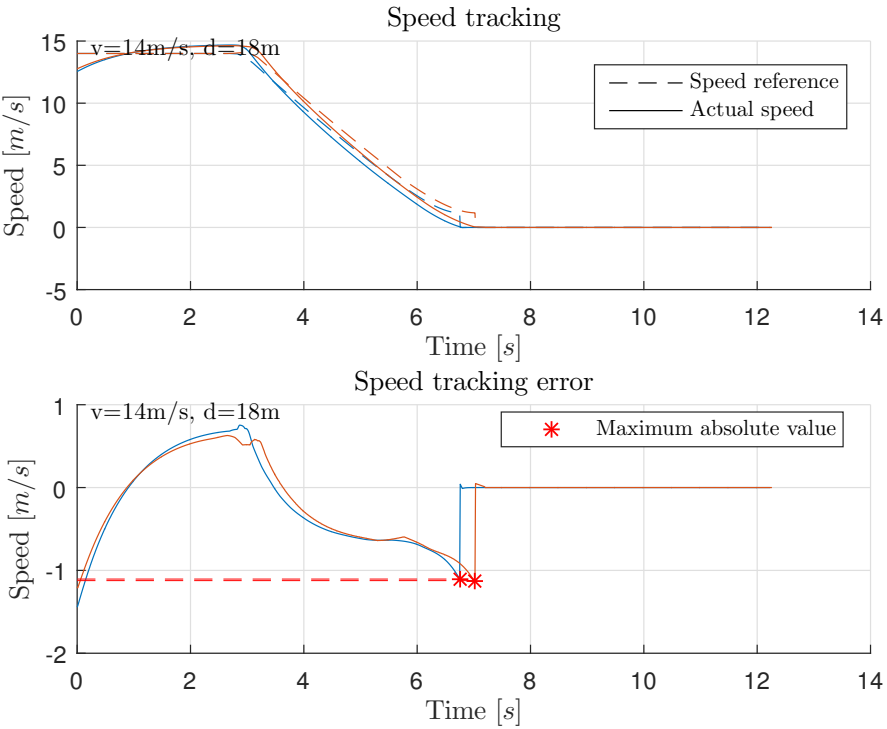

Figure A.18 Speed following and speed following error of the angle collision scenario with initial parameters $v = 14$m/s and $d = 18$m. Simulated on the driving robot simulator. The color code of the data in this figure is in unison with the colors of the data in the other figures of this test, which makes it easy to follow the data that corresponds to the different vehicles. Blue (Vehicle 1), red (Vehicle 2).

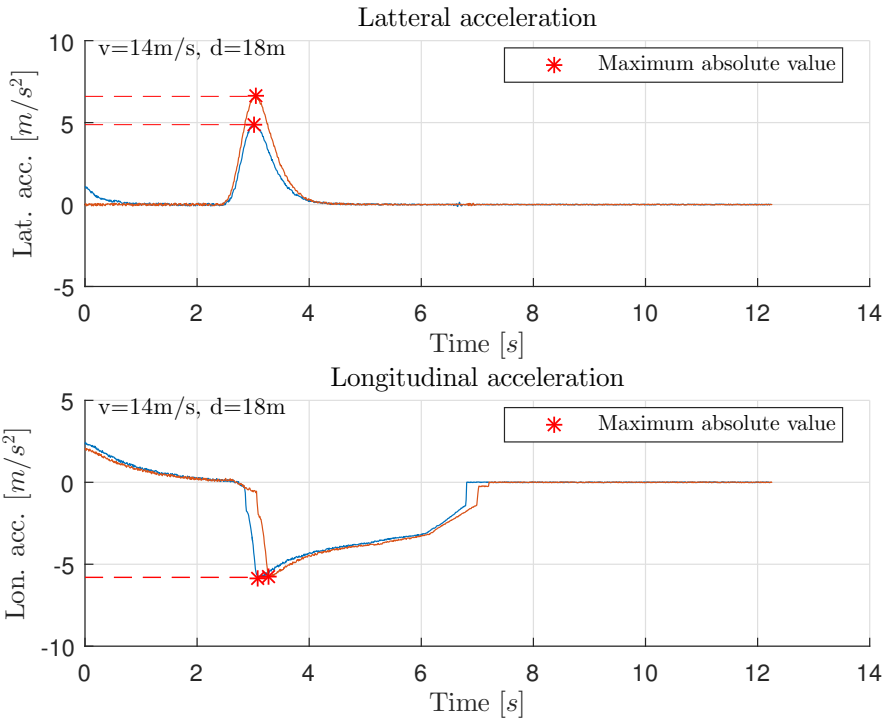

Figure A.19 Lateral and longitudinal accelerations of the angle collision scenario with initial parameters $v = 14$m/s and $d = 18$m. Simulated on the driving robot simulator. The color code of the data in this figure is in unison with the colors of the data in the other figures of this test, which makes it easy to follow the data that corresponds to the different vehicles. Blue (Vehicle 1), red (Vehicle 2).

B

Success map exemplified

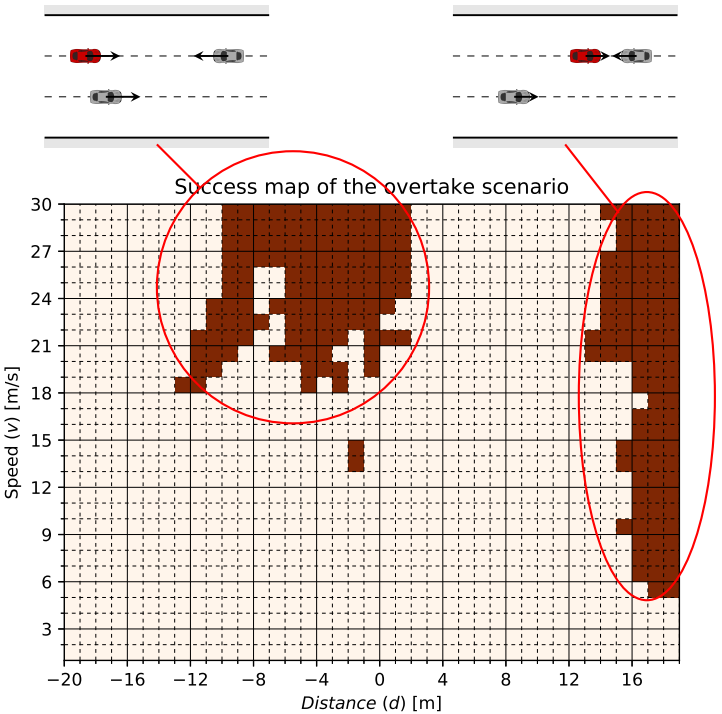

Figure B.1 Success map of the path generation of the overtake scenario. Red color symbolizes an unsuccessful trajectory generation of the B-ORCA algorithm, and white color symbolizes a successful trajectory generation. The values on the axes are the input parameters of the scenario, defined in table 4.5. Exemplified with bird's eye views of the scenario configurations of unsuccessful trajectory generations.

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER'S THESIS

Date of issue
June 2019

Document Number
TFRT- 6087

Author(s)
Daniel Johansson

Supervisor
Angel Molina Acosta, Volvo Cars
Anders Robertsson, Dept. of Automatic Control, Lund University, Sweden
Anders Rantzer, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle

Dynamic path planning for collision avoidance in a robotized framework for autonomous driving verification

Abstract

Self-driving vehicles is a highly anticipated technology for increasing the safety and efficiency of automotive transportation systems by removing the risk of human errors. Volvo Car Corporation is determined to produce vehicles of full autonomy and highest safety within the near future. To achieve this goal, Volvo Cars is in parallel with the development of autonomous vehicles setting up a sophisticated pipeline for verifying and testing the autonomous driving functions. This thesis revolves around the last step of this pipeline, by implementing and further developing an algorithm for collision avoidance in a robotized framework for verification of autonomous driving where the functions are tested on real vehicles on a test track.

The proposed algorithm used to achieve collision avoidance in the robotized test framework is the Bicycle Optimal Reciprocal Collision Avoidance (B-ORCA) algorithm. This algorithm uses a construct called Velocity Obstacle to predict imminent collisions between vehicles in a scenario and then calculates the optimal velocities to avoid the collisions in a collaborative manner.

To evaluate the performance of the algorithm, a set of experiments were performed on the driving robots that will be used in the testing framework, both in simulations and on a real vehicle. The results from these experiments show that the current implementation of the B-ORCA algorithm guarantees accurate safe trajectories up to speeds of 50km=h. To support speeds above 50km=h, the simple Kinematic bicycle model currently used to calculate the trajectories has to be replaced with a more sophisticated motion model. This new model has to better model the lateral acceleration that, with too high values, was shown to be the main parameter that made the vehicle not follow the safe trajectories as desired.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-98

Recipient's notes

Security classification

http://www.control.lth.se/publications/